SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2285

AUTENTIKACIJA OSOBA ANALIZOM IZGLEDA LICA

Ivana Babić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2285

AUTENTIKACIJA OSOBA ANALIZOM IZGLEDA LICA

Ivana Babić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2285

Pristupnica:	**Ivana Babić (0036478852)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Autentikacija osoba analizom izgleda lica**

Opis zadatka:

Raspoznavanje lica je važan problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme najbolji rezultati u tom području postižu se dubokim konvolucijskim modelima. Ovaj rad razmatra nadzirane pristupe gdje je svako lice iz skupa za učenje pridruženo odgovarajućoj osobi. U okviru rada, potrebno je proučiti konvolucijske arhitekture za raspoznavanje lica. Oblikovati klasifikacijski model te ga vrednovati na skupu LFW. Preporučiti smjernice za ugradnju modela u praktičan sustav za autentikaciju osoba. Validirati hiperparametre, prikazati i ocijeniti ostvarene rezultate te provesti usporedbu s rezultatima iz literature. Predložiti pravce budućeg razvoja. Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

Sadržaj

Uvod

Raspoznavanje lica (engl. *face recognition*) važan je problem računalnog vida s mnogim primjenama.
Posljednjih godina konvolucijske neuronske mreže (engl. *convolutional neural network, CNN*) pokazale su se jako uspješne u savladavanju problema raspoznavanja lica, prvenstveno zahvaljujući naprednim mrežnim arhitekturama.

Postignuća u razvoju kvalitetnih rješenja za problem raspoznavanja lica potaknula su razvoj sustava koji se temelje na takvim rješenjima, poput sustava za autentikaciju osoba s pomoću slike lica. Takvi sustavi spadaju u skupinu takozvanih sustava za biometrijsku autentikaciju (engl. *biometric authentication systems*).
Biometrija se bavi identifikacijom individualca na temelju nekih osobi jedinstvenih bioloških ili bihevioralnih karakteristika [1], poput otiska prsta, geometrije lica ili izgleda šarenice oka.
Autentikacija osobe raspoznavanjem lica smatra se jednom od najmanje intruzivnih biometrijskih metoda [2] jer, barem u teoriji, osoba mora samo približiti lice kameri i sustav na temelju slike lica dobivene kamerom obavi autentikaciju.

Međutim, koliko god se činila obećavajućom, tehnologija za autentikaciju osoba raspoznavanjem lica ima i svojih nedostataka. Duboki modeli naučeni rješavati problem raspoznavanja lica na temelju slike lica rade samo identifikaciju (ili verifikaciju) osobe. Slika lica koju takav model analizira može biti slika stvarne osobe koja u trenutku autentikacije fizički stoji ispred kamere, ali isto tako može biti i slika osobe prikazana kameri, na primjer, na zaslonu mobilnog telefona. Duboki model naučen samo za zadatak raspoznavanja lica iz tog će lica izvući značajke potrebne da se obavi verifikacija lica koje mu je prikazano, ali ne radi provjeru je li prikazano lice stvarna osoba koja stoji ispred kamere ili je potencijalni pokušaj prijevare.

Razvoj biometrijskih sustava za autentikaciju prati i aktivno proučavanje ranjivosti takvih sustava na napade lažiranjem (engl. *spoofing*) i traženje boljih i sigurnijih rješenja [3].

Sprječavanje lažiranja lica (engl. *face anti-spoofing*) postupak je provjere slike u svrhu utvrđivanja je li slika lica predstavljena autentikacijskom sustavu slika stvarne (žive) osobe ili je reproducirana ili sintetička, odnosno lažna.
Kada sustav za provjeru autentičnosti lica treba prepoznati lice osobe s pomoću kamere i pripadajućeg softvera za raspoznavanje slike lica, važno je biti siguran da osoba koja traži provjeru autentičnosti zapravo predstavlja svoje lice kameri u vremenu i mjestu zahtjeva za autentikaciju.
Suprotno tome, prevarant bi mogao pokušati predstaviti kameri masku, fotografiju ili videozapis koji sadrži lice legitimnog klijenta kako bi sustav, prepoznavši lice legitimnog klijenta ispred kamere, napadaču omogućio pristup sustavu kakav inače imaju samo autorizirani korisnici. Ta vrsta prijetnje autentikacijskim sustavima općenito je poznata kao napad ponovnog ponavljanja (engl. *replay attack*).

U sklopu ovog diplomskog rada istražene su arhitekture i funkcije gubitka dubokih neuronskih mreža za raspoznavanje lica, ranjivosti biometrijskih sustava za autentikaciju raspoznavanjem lica i potencijalna rješenja. Također je implementiran i sustav za autentikaciju korisnika s pomoću lica koji koristi neka od navedenih rješenja.

1. Umjetne neuronske mreže

Najjednostavniju definiciju umjetne neuronske mreže (engl. *artificial neural network, ANN*), dao je dr.Robert Hecht-Nielsen, izumitelj jednog od prvih neuroračunala. Neuronsku mrežu definirao je kao:

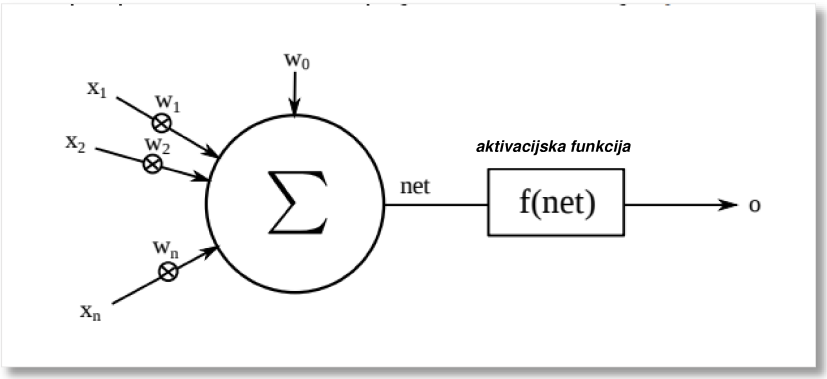
"računalni sustav sastavljen od određena broja jednostavnih, međusobno povezanih jedinica za obradu koje obrađuju informacije tako da na ulaze reagiraju dinamičkom promjenom stanja".

Ideja umjetnih neuronskih mreža inspirirana je strukturom središnjeg živčanog sustava sisavaca [10] i načinom na koji ljudi najbolje usvajaju nove informacije – učenjem iz primjera. Međutim, prvotni cilj da se takve mreže izrade i dovedu na razinu kompleksnosti i sofisticiranosti ljudskog mozga uskoro je napušten.

Umjesto toga, fokus je prebačen na izradu modela koji su specijalizirani za jedan specifičan zadatak i taj zadatak uspješno odrađuju. Ideja učenja iz primjera je zadržana. Neki od zadataka za koje se danas koriste umjetne neuronske mreže uključuju raspoznavanje objekata, identifikacija osoba na temelju lica, prepoznavanje govora, strojno prevođenje teksta, medicinska dijagnostika i mnoge druge.

1.1 Građa umjetne neuronske mreže

Umjetna neuronska mreža sastavljena je od jednostavnih, međusobno povezanih obradbenih jedinica koje se nazivaju neuroni. Svaki neuron prima određeni broj ulaznih signala koje onda obrađuje i rezultat obrade prosljeđuje dalje drugim neuronima u mreži s kojima je povezan. Slika 1.1 prikazuje građu umjetnog neurona.



Slika 1.1: Umjetni neuron prikazan McCulloch-Pitts modelom

Neuron formalno definiramo kao funkciju koja na ulazu prima vektor $x = [x_1, x_2, \dots, x_n]$ i preslikava ga u skalar o. Svaki ulazni parametar x_i množi se pripadajućom težinom w_i. Zatim svi umnošci $x_i * w_i$ zbroje i njihovom zbroju doda se vrijednost w_0 koja se naziva prag (engl. *bias*). Dakle, možemo pisati:

$$net = w_0 + \sum_{i=1}^{n} x_i * w_i.$$

Često se radi lakšeg zapisa definira $x_0 = 1$ i piše:

$$net = \sum_{i=0}^{n} x_i * w_i.$$

Nakon toga, dobivena suma produkata predaje se na ulaz aktivacijske funkcije f te se izlaz neurona računa kao: $o = f(net) = f(\sum_{i=0}^{n} x_i * w_i)$.

Prag je ekvivalentan pomicanju aktivacijske funkcije po apscisi. Dodavanjem slobode pomicanja po apscisi proširuje se područje djelovanja neurona, čime se mogu lakše modelirati ulazi neurona čije težište nije u ishodištu (uz pretpostavku aktivacijske funkcije koja je simetrična oko nule) [16].

Neuroni unutar mreže organizirani su u slojeve kao što je prikazani na Slika 1.2. Ulazni podatci predaju se mreži u ulaznom sloju (engl. *input layer*) koji komunicira s jednim ili više skrivenih slojeva (engl. *hidden layer*) u kojima se odvija stvarna obrada podataka preko sustava težinskih veza. Skriveni slojevi povezani su na izlazni sloj (engl. *output layer*). Zadaća skrivenih slojeva jest transformirati ulaze dobivene preko ulaznog sloja u nešto što izlazni sloj može koristiti za izražavanje konačnog (željenog) izlaza mreže.

Ovakva struktura neurona i cjelokupne mreže omogućava da mreža nauči preslikati podatke koje dobije na ulazu u željene izlaze.

Neka je stvarni odnos ulaza mreže x i željenog izlaza y izražen funkcijom $y = f^*(x)$.

Cilj je pronaći najbolju aproksimaciju te funkcije parametarskim modelom $\hat{y} = f(x, \Theta)$, gdje je Θ skup parametara (težina i pragova) koje mreža uči na podatcima.

Stvarni odnos između ulaza i izlaza najčešće nije linearan. Bez nelinearnih aktivacijskih funkcija, svaki neuron radio bi samo linearnu operaciju množenja ulaznog vektora s vektorom težina, a mreža bi onda bila kompozicija linearnih funkcija i samim time linearna funkcija. Zahvaljujući nelinearnim aktivacijskim funkcijama, mreža je sposobna naučiti kompleksnije, nelinearne odnose ulaza i željenih izlaza, a time i kvalitetniju aproksimaciju tražene funkcije f^*.

Neke od najpoznatijih aktivacijskih funkcija su ReLu (engl. *Rectified Linear Unit*) i sigmoidalna funkcija.

ReLU funkcija zadana je formulom

$$ReLU(x) = max(0, x),$$

a sigmoidalna funkcija formulom

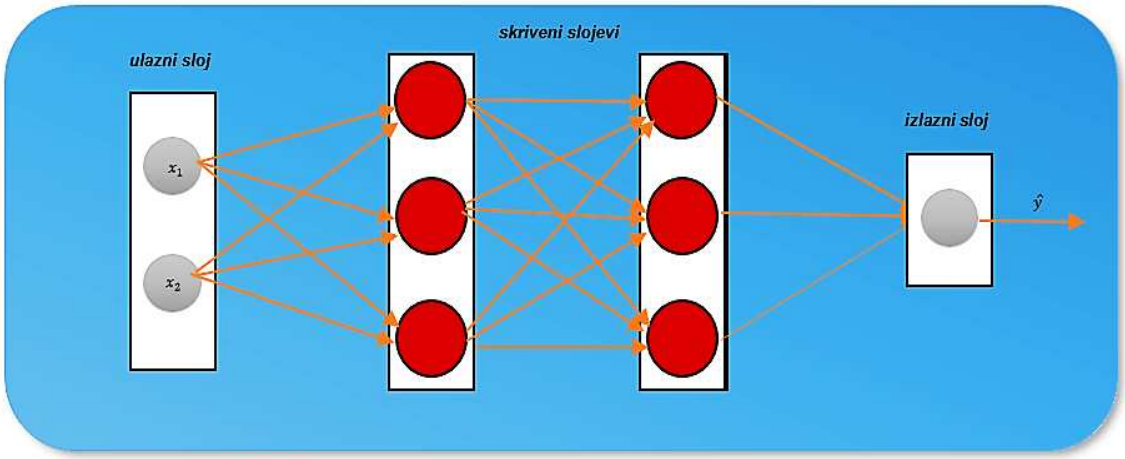
$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

1.2 Unaprijedne neuronske mreže

Unaprijedna neuronska mreža (engl. *feed- forward neural network*) vrsta je umjetne neuronske mreže u kojoj se informacija predana mreži na ulazu kreće, odnosno prosljeđuje kroz slojeve, samo u jednom smjeru, prema izlazu mreže. Dakle, mreža ne sadrži cikluse niti petlje u vezama među neuronima.

Osnovni oblik unaprijedne neuronske mreže jest potpuno povezana mreža (engl. *fully connected network*), što je lanac potpuno povezanih slojeva. To znači da je svaki neuron jednog sloja povezan sa svim neuronima sljedećeg sloja.

Slika 1.2 prikazuje potpuno povezanu mrežu koja ima ulazni sloj, 2 skrivena sloja i izlazni sloj. Strelice pokazuju smjer širenja informacije u mreži.



Slika 1.2: Unaprijedna potpuno povezana neuronska mreža

Umjetne neuronske mreže su univerzalni aproksimatori. Pojam dolazi iz matematičke teorije umjetnih neuronskih mreža i tiče se unaprijednih neuronskih mreža.
U tom kontekstu, teorem univerzalne aproksimacije [12] govori o tome da su unaprijedne neuronske mreže sposobne aproksimirati bilo koju mjerljivu kontinuiranu funkciju do željenog stupnja preciznosti.

2. Svojstva i građa konvolucijske neuronske mreže

Konvolucijske neuronske mreže, predstavljene prvi puta u radu [13] 1998.godine, vrsta su unaprijednih umjetnih neuronskih mreža koje su se pokazale jako uspješne u obradi podataka s topologijom rešetke, što je oblik strukture definiran relacijom susjedstva i može se prikazati matrično. Tipični primjeri takvih podataka su vremenski slijed (1D), siva slika (2D) ili slika u boji (3D).

Budući da su specijalizirane za procesiranje specifične vrste podataka i njihova struktura prilagođena je vrsti podataka koje obrađuju u odnosu na potpuno povezanu unaprijednu mrežu.
Tipična konvolucijska mreža sastoji se od sljedećih vrsta slojeva:

- konvolucijski sloj (engl. *convolutional layer*),
- sloj sažimanja (engl. *pooling layer*),
- ReLU sloj (engl. *ReLU layer; Rectified Linear Unit layer*),
- potpuno povezani sloj (engl. *fully connected layer*) i
- sloj gubitka (engl. *loss layer*).

Broj i raspored ovih slojeva može varirati.

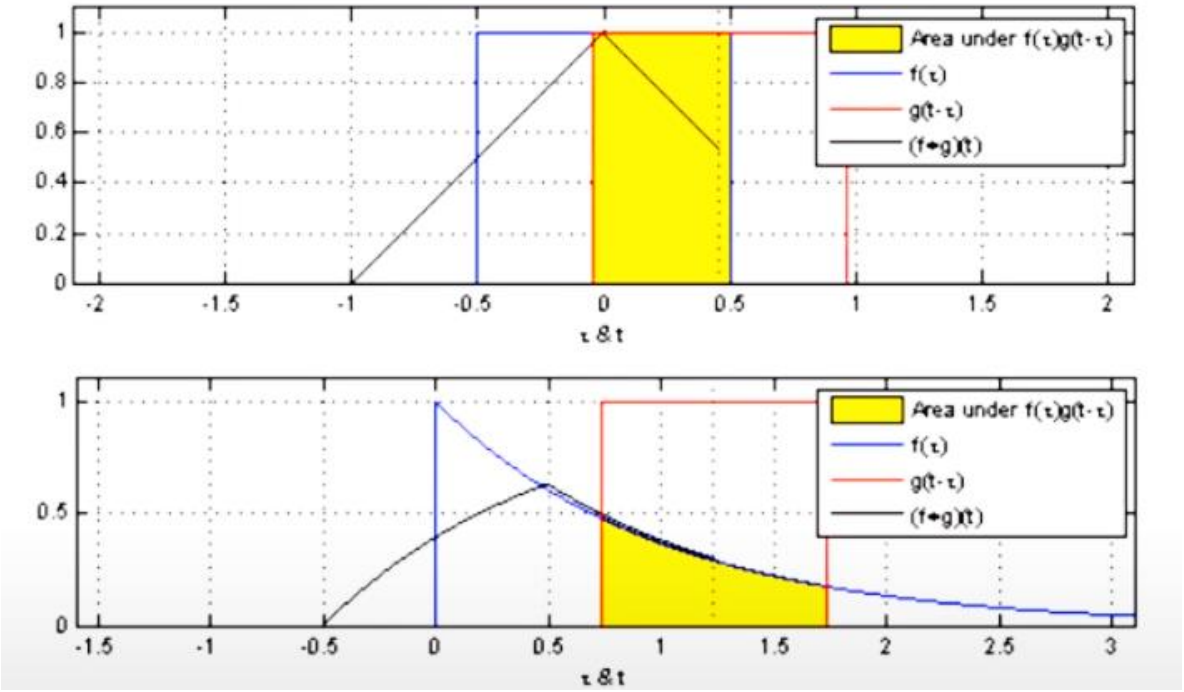
Ako mreža na ulazu dobiva RGB slike veličine 200x200 piksela ta slika je zapravo 3D matrica dimenzija 200x200x3 (gdje je 3 broj kanala boje, RGB). U tom slučaju, jedan neuron u prvom skrivenom sloju potpuno povezane mreže morao bi imati 200x200x3=120 000 težina, po jednu za svaku vrijednost elementa slike (piksela) u svim kanalima (RGB). Budući da bi sigurno htjeli više takvih neurona u sloju, ali i više slojeva u mreži, ukupan broj parametara bi jako brzo eskalirao. Prevelik broj parametara dovodi do sporijeg učenja mreže, potrebe za većim brojem primjera za učenje, ali i do problema prenaučenosti mreža. Konvolucijske neuronske mreže posjeduju 2 svojstva kojima smanjuju ukupan broj parametara u mreži:

1. Raspršena povezanost među slojevima (engl. *sparse interactions between layers*),
2. Dijeljenje parametara (engl. *parameter sharing*).

Ova svojstva bit će detaljnije objašnjena u sljedećim poglavljima o konvolucijskom sloju.

2.1 Konvolucijski sloj

Konvolucijski sloj je elementarni dio konvolucijske mreže koji zapravo odrađuje većinu posla u mreži. Naziv dolazi od operacije konvolucije koja se obavlja unutar sloja. Operacija konvolucije definirana je nad dvije funkcije realne ili kompleksne varijable te kao izlaz daje treću funkciju koja je definirana kao količina preklapanja između prve funkcije te okrenute i pomaknute druge funkcije [15].



Slika 2.1: Konvolucija funkcija f(t) i g(τ - t)[1]

Konvolucija realne neprekinute funkcije definirana je izrazom:

$$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)d\tau.$$

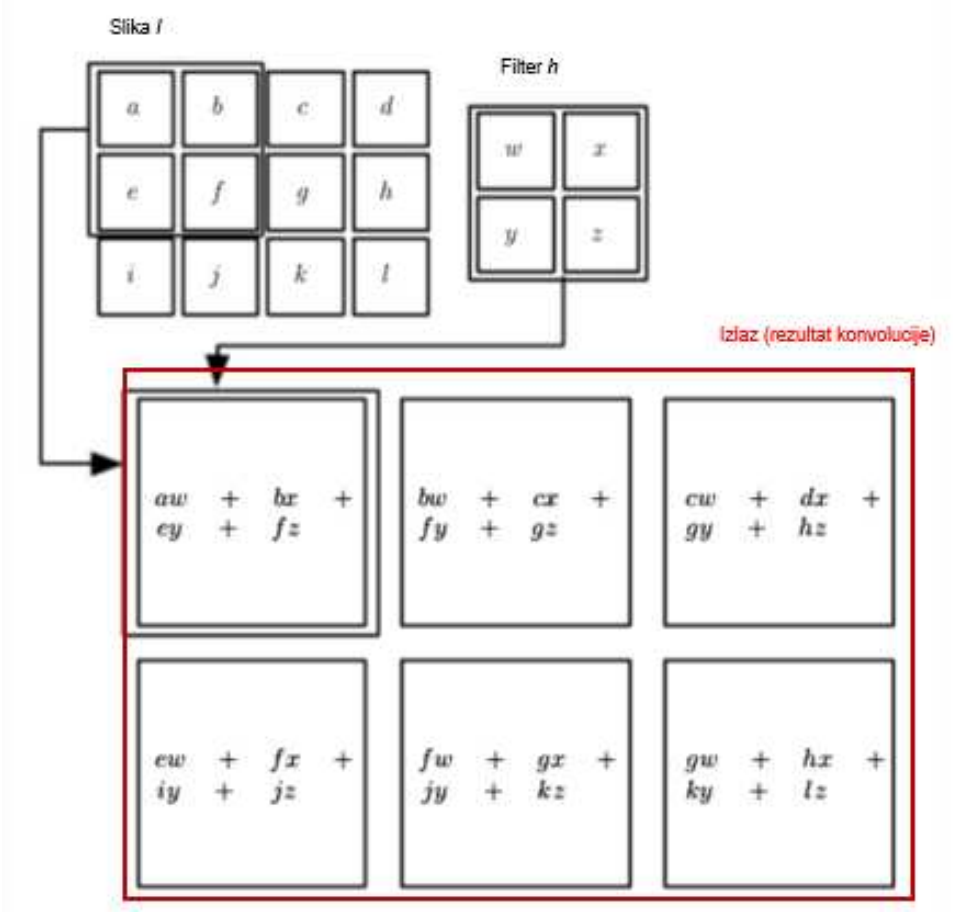
Formula predstavlja količinu površine filtera *g* koja preklapa funkciju *f* u trenutku *τ* tokom cjelokupnog vremena *t*. Na Slika 2.1, graf funkcije *f* prikazan je plavom bojom, graf funkcije *g* crvenom bojom, a površina preklapanja filtera *g* i funkcije *f* u danom trenutku *τ* obojana je žutom bojom.

[1] Slika preuzeta s adrese: https://cutt.ly/myMsnQD

U slučaju da na ulazu u mrežu imamo višedimenzionalni podatak (poput sive slike) potrebna je i višedimenzionalna fukncija *g*, odnosno višedimenzionalni filter *g*.
Neka je *I* siva slika i *h* filter koji će se koristiti za operaciju konvolucije nad slikom. Tada se rezultat konvolucije dobije tako da filter „klizi“ po slici duž dimenzija širine i dužine. Takva operacija konvolucije definirana je formulom:

$$(I * h)(x, y) = \int_0^x \int_0^y I(x - i, y - j)h(i, j)\, di\, dj.$$

Slika 2.2 prikazuje pojednostavljen grafički prikaz operacije konvolucije nad slikom *I*.

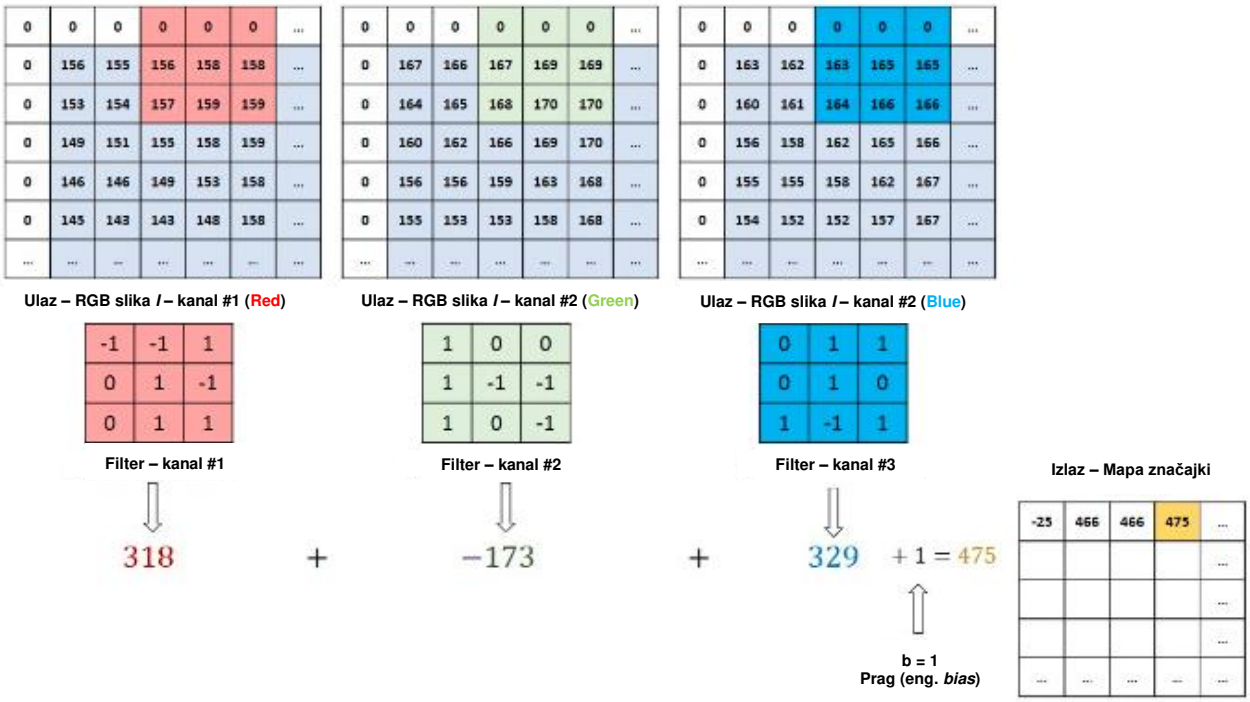


Slika 2.2: Operacija konvolucije nad 2D ulazom

Dimenzije slike *I* na Slika 2.2 su 4x3, dimenzija filtera *h* je 2x2, korak pomaka filtera po slici je (1,1), a dimenzija izlaza je 3x2. Izlaz se još naziva i **mapa značajki.** Vrijednosti ulaznog dijela unutar prozora se množe s filtrom i sumiraju te daju izlaze za pojedine dijelove izlazne mape značajki. Filter se primjenjuje tako da se pomiče kroz cijeli ulaz. Rezultat konvolucije je dvodimenzionalna mapa značajki koja predstavlja odziv filtera

na svakoj prostornoj poziciji. Zapravo, mreža će naučiti težine unutar filtera kako bi se filter aktivirao na mjestima gdje prepoznaje određena slikovna svojstva, na primjer, određene vrste rubova.

U slučaju 3D ulaza poput RGB slike, svaki filter u sloju također mora biti 3D.



Slika 2.3: Operacija konvolucije nad 3D ulazom[2]

Slika 2.3 prikazije operaciju konvolucije nad 3D ulazom. Filter ima dimenziju 3x3x3. Klizeći po slici filter radi operaciju konvolucije nad svakim 2D kanalom s odgovarajućim 2D dijelom filtera i dobiju se 3 međurezultata. Zbroj ta 3 međurezultata i praga daje rezultat koji će se upisati na odgovarajuće mjesto izlazne mape značajki. Dakle, **konvolucija 3D filterima nad 3D ulazom daje 2D mapu značajki koja predstavlja odziv filtera na svakoj prostornoj poziciji.**

[2] Slika preuzeta s adrese: https://cutt.ly/PyMgitl

2.1.1 Hiperparametri konvolucijskog sloja i veličina izlaza

Parametri se definiraju kao dijelovi modela koji uče iz podataka. Parametri konvolucijskog sloja (težine i prag) nalaze se u filterima. Dakle, tokom učenja mreže vrijednosti unutar filtera se modificiraju.

Hiperparametri sloja opisuju konfiguraciju sloja. Njihove vrijednosti postavljaju se pri izradi modela. Hiperparametri konvolucijskog sloja koji kontroliraju veličinu izlaznog volumena konvolucijskog sloja su:

- korak pomaka filtera po širini/visini prilikom konvolucije (engl. *stride*) - veći korak filtera znači manje dimenzije izlaznog volumena;
- popunjavanje nulama (engl. *zero-padding*):
  popunjavanje ulaznog volumena nulama u području oko rubova koje omogućava kontrolu nad prostornom veličinom izlaznog volumena (najčešće se koristi da bi se očuvale dimenzije ulaza, tako da visina i širina ulaza i izlaza budu jednake – „SAME“);
- veličina filtera kojim se obavlja konvolucija - obično se za filter bira kvadratna matrica veličine 3x3, 5x5 ili 7x7;
- broj filtera u sloju.

Dimenzionalnost ulaza u konvolucijski sloj označit ćemo sa $W_1 x H_1 x D_1$.
Hiperparametri sloja su:

- broj filtera K
- dimenzionalnost filtera FxF
- korak pomaka jezgre pri konvoluciji (engl.*stride*) S
- količina popunjavanja nulama (engl.*zero-padding*) P

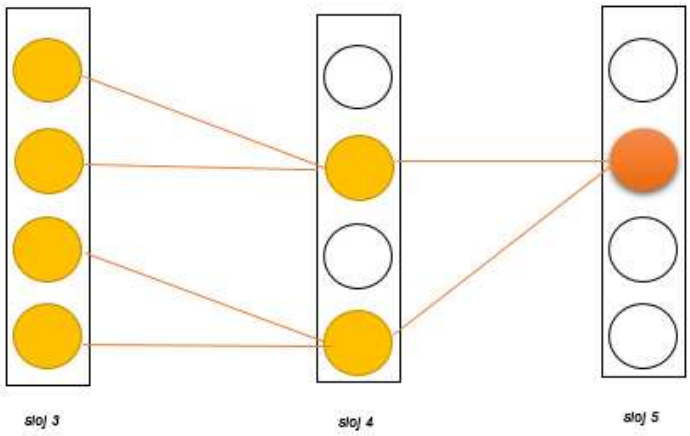
Na izlazu iz sloja nastaje volumen dimenzija $W_2 x H_2 x D_2$ koji se dobije na sljedeći način:

$$W_2 = \frac{W_1 - F + 2*P}{S} + 1, \qquad H_2 = \frac{H_1 - F + 2*P}{S} + 1, \qquad D_2 = K.$$

2.1.2 Raspršena povezanost među slojevima

U klasičnoj potpuno povezanoj mreži svaki neuron jednog sloja povezan je sa svim neuronima sljedećeg sloja. Svaka od tih veza predstavlja jedan parametar koji mreža mora naučiti. Budući da je broj parametara u takvim mrežama jako velik, za učenje modela potrebno je više podataka, memorije i vremena. Prevelik broj parametara dovodi i do prenaučenosti modela [14].

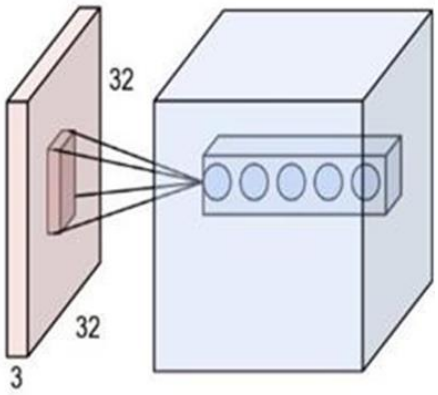
Konvolucijske neuronske mreže smanjuju broj potrebnih parametara u mreži pomoću indirektnih interakcija među neuronima.



Slika 2.4: Primjer - raspršena povezanost

Na Slika 2.4, označeni neuron iz sloja 5 nije povezan sa svim neuronima iz sloja 4, već samo sa 2 takva neurona. Ta 2 neurona iz sloja 4 čine **receptivno polje** narančastog neurona iz sloja 5 i oni direktno utječu na neuron iz sloja 5, jer njihov izlaz čini ulaz spomenutog neurona u sloju 5. Nadalje, svaki od tih neurona iz sloja 4 ima svoja receptivna polja u sloju 3 i tako dalje. Ovakva povezanost neurona iz različitih slojeva naziva se **raspršena povezanost**. Pokazalo se da je u dubljim mrežama raspršena povezanost sasvim dovoljna za propagaciju informacije kroz mrežu. Također, zbog manjeg broja veza smanjuje se i broj parametara koje mreža mora naučiti u odnosu na potpuno povezanu mrežu.

U konvolucijskom sloju svaki neuron povezan je samo sa lokalnom regijom ulaza. Ako je u mreži prvi skriveni sloj konvolucijski sloj, onda će zbog raspršene povezanosti svaki neuron konvolucijskog sloja za ulaz primiti malu, lokalnu grupu (susjednih) piksela slike, a ne sve piksele slike. Prostorni opseg ove veze je hiperparametar sloja i naziva se **receptivno polje neurona** (engl. *receptive field)* te odgovara veličini filtera. Važno je naglasiti asimetriju u načinu na koji se odnosimo prema dimenzijama širine i visine u odnosu na dubinu - veze su lokalne (raspršene) u prostoru duž osi širine i visine, ali su uvijek potpune duž cijele dubine ulaza.



Slika 2.5: Dimenzionalnost izlaza

Mape značajki dobivene pojedinom filterima nanizane duž osi dubine tvore izlaz konvolucijskog sloja. Označeni dio izlaznog volumena na Slika 2.5 čine rezultati konvolucije različitim filterima na istom dijelu slike. Jedan od filtera je na tom dijelu slike možda detektirao vertikalni rub, neki drugi horizontalni rub, itd. Budući da su elementi filtera parametri koje mreža uči, teško je biti siguran koji se filter za što specijalizirao.

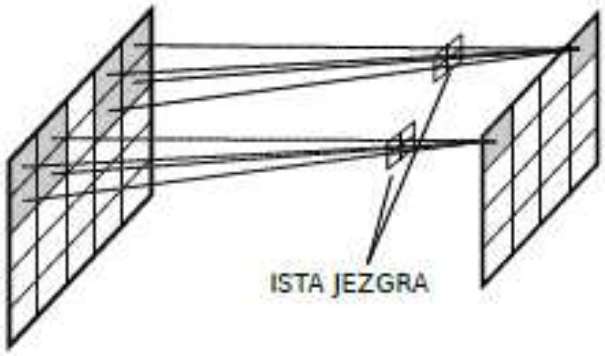
Korištenje raspršene povezanosti može uvelike pomoći u učenju značajki ukoliko je skup za učenje takav da ima težnju konvergiranju istom manjem broju značajki. U slučaju potpune povezanosti, sve mape primaju sve vrijednosti iz prethodnog sloja pa je moguće da dođe do situacije da dvije ili više mapa konvergiraju k istoj vrijednosti, odnosno da se dva ili više filtera specijaliziraju za iste značajke. Uvođenjem raspršene povezanosti mape dobivaju različite ulaze čime se osigurava konvergencija k različitim vrijednostima.

2.1.3 Dijeljenje parametara

Dijeljenje težina u konvolucijskim slojevim omogućava kontrolu nad brojem parametara u mreži. Broj parametara se može značajno smanjiti ako se sloj modelira imajući na umu jednu pretpostavku:

> *„Ako je korisno računati jednu značajku na poziciji (x_1,y_1), onda je vjerojatno da ju je korisno računati i na nekoj drugoj poziciji (x_2,y_2).“*

Drugim riječima, želimo ograničiti neurone u svakom 2D-sloju dubine ulaznog volumena (npr. volumen veličine 24x24x3 ima 3 sloja dubine) tako da koriste iste težine i pragove. Ako svi neuroni na istoj dubini koriste iste težine, onda se svaki 2D dio (engl. *slice*) izlaznog volumena računa kao konvolucija između ulaznog volumena i tih težina. Taj set težina naziva se filter ili jezgra (engl. *kernel*).



Slika 2.6: Dijeljenje parametara

Dijeljenje parametara omogućava da mreža nauči relevantne i diskriminativne značajke. Jezgre se specijaliziraju za određenu funkciju te postaju slične, na primjer, Haarovim i drugim značajkama. Bez dijeljenja parametara dijelovi neuronske mreže bi se prenaučili na određeni detalj podataka. S dijeljenjem na istu jezgru dolaze različiti podatci što povećava općenitost naučene značajke i poboljšava generalizacijske sposobnosti mreže [16].

2.2 Sloj sažimanja

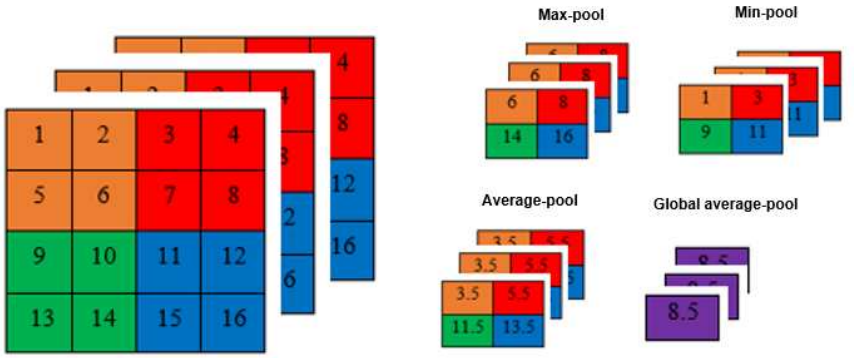
Postupak sažimanja (engl. *pooling*) smanjuje rezoluciju mapi značajki, povećava prostornu invarijantnost neuronske mreže (neosjetljivost na manje pomake značajki u uzorku/slici) te smanjuje vjerojatnost da se mreža prenauči [17].
Postupak se provodi tako da se *FxF* prostorno bliskih značajki mapira na jednu značajku na izlazu, gdje je *F* hiperparametar sloja sažimanja i označava veličinu regije (prozora) sažimanja. Sloj sažimanja također ima i hiperparametar *S* (engl. *stride*) koji označava korak pomaka prozora sažimanja po ulaznom volumenu, slično kao i kod konvolucijskog sloja.

Smanjenje rezolucije postiže se postepeno kroz više slojeva u mreži. To znači da u kasnijim slojevima mreža uči međusobni položaj značajki detektiranih u prethodnim slojevima (npr. položaj očiju, nosa i usta kod detekcije lica), ali zbog postupka sažimanja postaje otporna na manje varijacije u pomaku.
To svojstvo konvolucijskih neuronskih mreža naziva se **translacijska invarijantnost ili invarijantnost na pomak**. Invarijantnost na pomak je vrlo korisna ako pretpostavljamo da nam je za raspoznavanje bitnije detektirati prisutnost određene značajke, nego njihovu točnu lokaciju. Veličina regije sažimanja regulira dozu invarijantnosti: veća regija implicira invarijantnost na veće pomake.

Postoji više metoda kojima se može obaviti postupak sažimanja. Neke od često korištenih su:

- sažimanje maksimalnom vrijednošću (engl. *max-pooling*) – iz *FxF* prostorno bliskih značajki odabire se ona koja je najveća,
- sažimanje srednjom vrijednošću (engl. *average-pooling*) - iz *FxF* prostorno bliskih značajki rezultat se računa kao srednja vrijednost odabranih vrijednosti,
- sažimanje minimalnom vrijednošću (engl. *min*-pooling) – iz *FxF* prostorno bliskih značajki odabire se ona koja je najmanja,
- globalno sažimanje srednjom vrijednošću (engl. *global average-pooling*) – iz

FxF prostorno bliskih značajki rezultat se računa kao srednja vrijednost odabranih vrijednosti, s tim da prozor sažimanja *FxF* prekriva cijeli ulaz duž dimenzija širine i visine.



Slika 2.7: Princip rada različitih metoda sažimanja.

Slika 2.7 prikazuje način rada spomenutih metoda sažimanja.
Ulazni volumen ima dimenzije 4x4x3, gdje su radi jednostavnosti svi 2D dijelovi duž dimenzije dubine jednaki. Radimo sažimanje prozorom dimenzija 2x2 koji pomičemo korakom 2. Sažimanje se obavlja zasebno za svaki 2D dio duž dimenzije dubine.

Sažimanje maksimalnom vrijednošću – u početku prozor sažimanja prekriva prvu gornju četvrtinu ulaza. Iz tog dijela ulaza odabire maksimalnu vrijednost (6) i prosljeđuje kao izlaz. Zatim se prozor pomakne za 2 mjesta (na drugu gornju četvrtinu ulaza) i iz tog dijela ulaza opet odabire maksimalnu vrijednost (8), itd.

Sažimanje minimalnom vrijednošću – u početku prozor sažimanja prekriva prvu gornju četvrtinu ulaza. Iz tog dijela ulaza odabire minimalnu vrijednost (1) i prosljeđuje kao izlaz. Zatim se prozor pomakne za 2 mjesta (na drugu gornju četvrtinu ulaza) i iz tog dijela ulaza opet odabire minimalnu vrijednost (3), itd.

Sažimanje srednjom vrijednošću – u početku prozor sažimanja prekriva prvu gornju četvrtinu ulaza. Izračuna se srednja vrijednost elemenata iz tog dijela ulaza i

prosljeđuje kao izlaz. Zatim se prozor pomakne za 2 mjesta (na drugu gornju četvrtinu ulaza) i iz tog dijela ulaza se opet izračuna srednja vrijednost i proslijedi kao izlaz, itd.

Globalno sažimanje srednjom vrijednošću – u ovom slučaju prozor sažimanja prekriva cijeli 2D dio ulaza (duž dimenzija širine i visine). Izračuna se srednja vrijednost elemenata iz tog dijela ulaza i prosljeđuje kao izlaz.

Neka je $W_1 x H_1 x D_1$ dimenzionalnost ulaza u sloj sažimanja, FxF veličina prozora sažimanja i S korak pomaka prozora sažimanja po ulazu.
Za klasične operacije sažimanja (max-pool, min-pool, average-pool) dimenzija izlaza jednaka je$W_2 x H_2 x D_2$, gdje su:

$$W_2 = \frac{W_1 - F}{S} + 1, \qquad H_2 = \frac{H_1 - F}{S} + 1, \qquad D_2 = D_1.$$

Za operacije globalnog sažimanja dimenzija izlaza jednaka je:

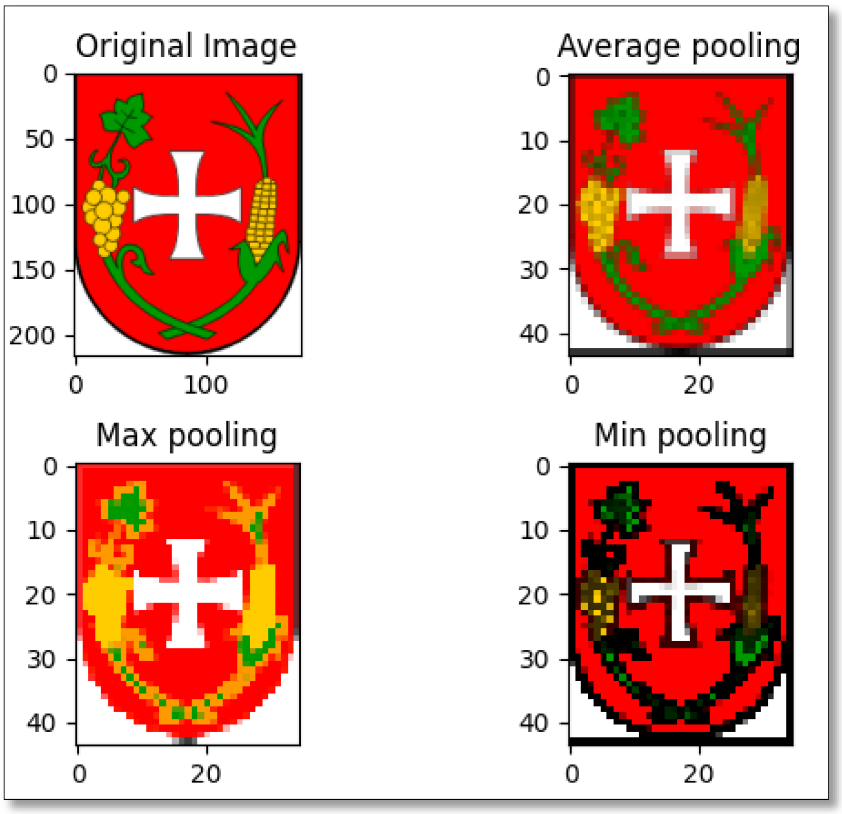
$$1 \; x \; 1 \; x \; D_1.$$

Operacija globalnog sažimanja najčešće se koristi na kraju mreže umjesto potpuno povezanog sloja. Ideja je generiranje mapi značajki pomoću niza konvolucijskih slojeva i slojeva sažimanja tako da se na kraju mreže dobiju mape značajki od kojih svaka odgovara jednoj kategoriji (semantičkoj klasi) klasifikacije [22]. Nakon toga, operacija globalnog sažimanja srednjom vrijednošću izračuna srednju vrijednost svake od tih mapi značajki i rezultat se proslijedi izlaznom sloju.

Izbor operacije sažimanja ovisi o podatcima na kojima mreža uči. Primjerice, sažimanje srednjom vrijednosti zagladi sliku, što smanjuje mogućnost detektiranja oštrijih elemenata slike.
Sažimanje maksimalnom vrijednošću odabire najsvjetlije piksele slike (one sa najvećim vrijednostima). Ono je korisno u situacijama kada je pozadina slike tamnija, a zanimljive značajke slike se onda nalaze u svjetlijim predjelima slike.
Sažimanje minimalnom vrijednošću odabire najtamnije piksele slike.

Ako radimo sa MNIST skupom podataka koji sadrži slike rukom pisanih znamenki (bijele znamenke na crnoj pozadini) najbolji izbor operacije sažimanja u sloju sažimanja bilo bi sažimanje maksimalnom vrijednošću (max-pool).

Slika 2.8 ilustrira rezultate različitih operacija sažimanja nad istom originalnom RGB slikom.



Slika 2.8: Rezultati različitih operacija sažimanja nad istom originalnom slikom

2.3 Sloj gubitka

Sloj gubitka određuje na koji način će mreža sankcionirati odstupanja između očekivanog i stvarnog izlaza mreže, što se obavlja pomoću funkcije gubitka (engl. *loss function*).

Postoje različite funkcije gubitka ovisno o zadatku koji mreža treba obavljati. Ovdje će biti objašnjeni gubitci koji se najčešće koriste kod klasifikacijskih zadataka i u modelima za raspoznavanje lica koji su korišteni u implementacijama u sklopu ovog rada.

Najprije je potrebno definirati aktivaciju izlaznog sloja mreže. Korištena je tzv. softmax funkcija koja se koristi kada je potrebno izlaz mreže klasificirati u jednu od *K* međusobno isključivih klasa.

Softmax funkcija ili normalizirana eksponencijala je generalizacija logističke funkcije koja pretvara *K*-dimenzionalni vektor z, čiji su elementi bilo koji realni brojevi, u *K*-dimenzionalni vekto $\sigma(z)$ čiji su elementi realni brojevi iz intervala [0,1], takvi da je zbroj elemenata vektora $\sigma(z)$ jednak 1.

Softmax funkcija dana je izrazom:

$$\sigma(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \qquad za\ j = 1,2,\dots,K.$$

Dakle, softmax kao izlaz daje normalizirane vjerojatnosti klasa, što znači sljedeće:
ako je izlaz softmax funkcije vektor duljine *K* i svaki element tog vektora pridjeljen je nekoj od *K* klasa, numerička vrijednost koja se nalazi na nekom indeksu *i* predstavlja vjerojatnost da uzorak predan mreži na klasifikaciju pripada klasi pridjeljenoj indeksu *i*.

Izlaz softmax funkcije prosljeđuje se funkciji koja računa gubitak, odnosno uspoređuje izlaz mreže sa točnim (očekivanim) izlazom te računa grešku.

2.3.1 Kategorička unakrsna entropija

Kategorička unakrsna entropija je gubitak korišten u modelima za višeklasnu klasifikaciju u *K* klasa. Koristi se i naziv **softmax gubitak**. Računa se tako da se izlaz softmax aktivacije proslijedi kao ulaz funkcije za unakrsnu entropiju.

Gubitak kategoričke unakrsne entropije definiran je formulom:

$$CE = -\sum_{i=1}^{K} p_i * log(q_i),$$

gdje *p* predstavlja distribuciju vjerojatnosti očekivanog (točnog) izlaza, a *q* procjenjene vjerojatnosti klasa koje je dala mreža na izlazu, odnosno:

- *p* = [0,...,1,...,0] , tj. ako uzorak predan mreži pripada klasi *i*, u vektoru *p* indeks dodjeljen klasi *i* imat će vrijednost (vjerojatnost klase *i*) 1, dok će sve ostale klase imati vjerojatnosti 0.
- *q* = *σ(z)* je izlaz softmax funkcije.

Označimo element vektora *p* koji ima vrijednost 1 sa p_i. Vrijednost izlaza softmax funkcije na odgovarajućem indexu onda iznosi:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}.$$

Tada formulu za gubitak kategoričke unakrsne entropije možemo kraće pisati kao:

$$CE = -log\left(\frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}\right),$$

budući da će svi ostali članovi sume biti jednaki 0, jer vrijedi da je $p_j = 0\ za\ j \neq i$.

Gubitak kategoričke unakrsne entropije korišten je u modelu izrađenom u sklopu ovog rada koji je učen za zadatak raspoznavanja ekspresija lica.

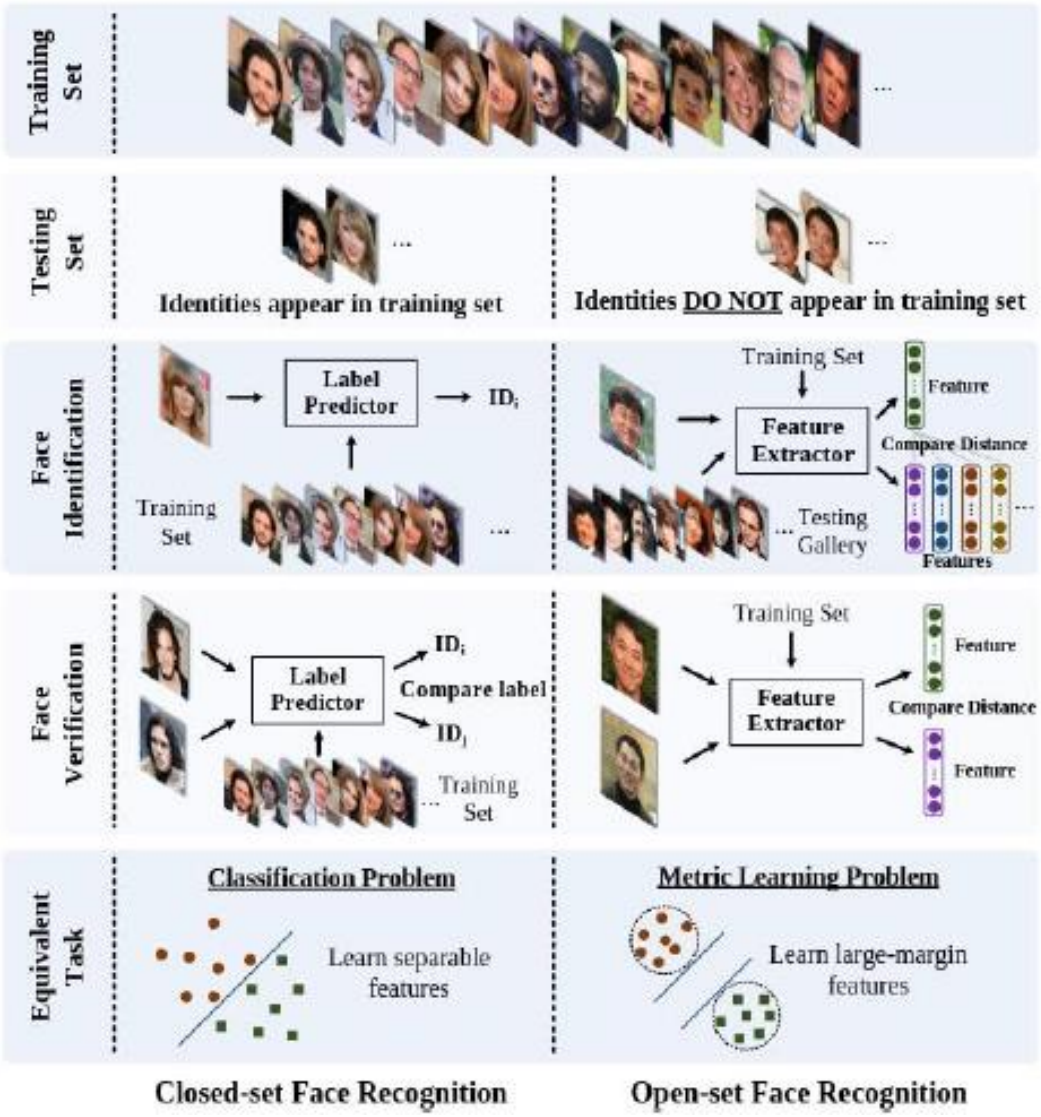
2.3.2 Gubitci u modelima za raspoznavanje lica s otvorenim skupom

Raspoznavanje lica obično se može podijeliti u dvije kategorije:

1. identifikacija lica (engl. *face identification*)
2. verifikacija lica (engl. *face verification*).

Identifikacija pridružuje lice specifičnom identitetu, a verifikacija određuje pripada li **par lica** istom identitetu.

U kontekstu testiranja raspoznavanje lica može se ocjenjivati (evaluirati) koristeći zatvoreni ili otvoreni skup, kako prikazuje Slika 2.9.



Slika 2.9: Usporedba postupka evaluacije modela za raspoznavanje lica sa zatvorenim i otvorenim skupom[3]

U zatvorenom skupu svi identiteti za testiranje unaprijed su definirani u skupu za učenje. Problemi raspoznavanja lica sa zatvorenim skupom mogu se uspješno riješiti

[3] Slika preuzeta iz [19]

izgradnjom klasifikatora koji će klasificirati slike (lica) testnog skupa u klase koje su zapravo identiteti iz skupa za učenje.

Kod problema s otvorenim skupom, identiteti za testiranje obično se odvajaju od skupa za učenje. Dakle, skup za testiranje ne sadrži niti jednu sliku osobe čije se slike pojavljuju u skupu za učenje . Kako je nemoguće klasificirati lica iz skupa za testiranje na one poznate identitete iz skupa za učenje, trebamo preslikati lica u diskriminativni prostor značajki. U ovom scenariju raspoznavanje lica može može se promatrati kao usporedba lica između onog koje analiziramo i svakog identiteta u galeriji (desna strana Slika 2.9).

2.3.2.1 Funkcije gubitka za problem raspoznavanja lica

Reprezentacija lica pomoću vektora značajki dobivenog pomoću konvolucijske neuronske mreže pokazala se kao najbolja metoda reprezentacije lica za probleme verifikacije lica i raspoznavanja lica [23].
Raspoznavanje lica s otvorenim skupom u osnovi je problem učenja diskriminativnih značajki s velikom marginom. Očekuje se da će željene značajke za raspoznavanje lica s otvorenim skupom zadovoljavati kriterij da je maksimalna udaljenost unutar klase manja od minimalne udaljenosti među klasama pod određenim metričkim prostorom. Učenje značajki s ovim kriterijem općenito je teško zbog velikih varijacija unutar jedne klase i visoke međusobne sličnosti različitih klasa [19].

Nekoliko pristupa temeljenih na konvolucijskim neuronskim mrežama uspijevaju učinkovito formulirati gore spomenuti kriterij u funkcijama gubitaka.

Zahvaljujući naprednim mrežnim arhitekturama i jako velikim skupovima za učenje, funkcije gubitka kao što su softmax gubitak ili trojni gubitak (engl. *triplet* loss) daju jako dobre rezultate na problemima raspoznavanja lica. Međutim, i softmax gubitak i trojni gubitak imaju određene mane.

Značajke naučene modelom sa softmax gubitkom dovoljno su odvojive za problem klasifikacije zatvorenog skupa, ali ne i dovoljno diskriminativne za problem raspoznavanja lica s otvorenim skupom.
Problemi trojnog gubitka su pronalazak kvalitetne metode generiranja trojki, ali i mogućnost kombinatorne eksplozije u broju trojki, posebno za velike skupove podataka, što dovodi do značajnog porasta broja iteracija učenja.

SphereFace rad [19] predstavlja novu ideju kutne margine i novu funkciju gubitka nazvanu A-Softmax (engl. *angular softmax*). Temelji se na pretpostavci da se matrica linearnih transformacija iz zadnjeg potpuno povezanog sloja neuronske mreže može koristiti za reprezentaciju centara klasa u kutnom prostoru (engl. *angular space*) [20]. A-softmax smanjuje kut između naučenih značajki i pripadnih težina množeći ga s konstantom *m* i računajući kosinus tako dobivenog kuta, čime se povećava kompaktnost uzoraka iste klase i veličina margine između različitih klasa. ArcFace rad [20] predlaže novu funkciju gubitka koja se temelji na sličnim pretpostavkama kao i Sphereface te rješava problem nestabilnosti tokom učenja koji je često bio prisutan u modelima sa A-softmax gubitkom.

2.3.2.2 Definicija A-softmax i ArcFace gubitaka

Funkcije gubitka poput softmaxa temelje se na pronalaženju Euklidske margine među naučenim značajkama. Međutim, značajke naučene pomoću softmax gubitka imaju intrinzičnu kutnu (engl. *angular*) distribuciju.
Softmax gubitak definiran je sljedećom jednadžbom:

$$L_1 = \frac{-1}{N}\sum_{i=1}^{N} log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}$$

gdje su $x_i \in R^d$ duboke značajke i-tog primjera koje pripadaju y_i-toj klasi. Korespondentni ugrađeni vektor značajki (engl. *embedding feature vector*) je dimenzije $d = 512$.

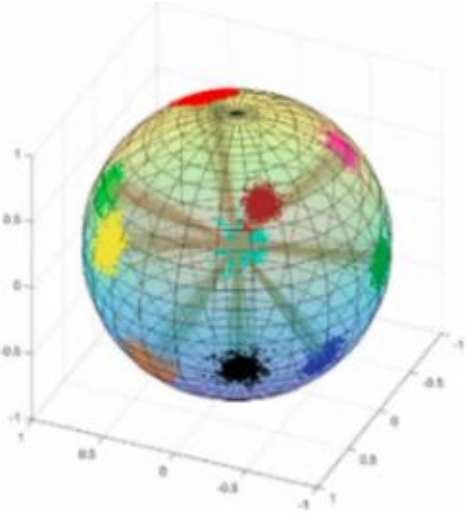
$W_j \in R^d$ je $j-ti$ stupac težina $W \in R^{dxn}$, a $b_j \in R^n$ pripadni prag. Veličina mini-grupe za učenje (engl. *mini-batch size*) je N, a broj klasa n.

Iako je softmax često korištena funkcija gubitka, ona ne optimizira eksplicitno ugrađene vektore značajki u svrhu poticanja veće sličnosti između primjera iste klase i veće različitosti između primjera različitih klasa, što rezultira lošijim performansama na skupovima slika gdje postoje velike varijacije unutar klase (npr. varijacije u pozi ili slike iste osobe u različitoj životnoj dobi) [20].
Radi jednostavnosti, pretpostavit ćemo da su svi pragovi $b_j = 0$. Onda možemo pisati da je

$$W_j^T x_i + b_j = W_j^T x_i = \|W_j\| \|x_i\| cos\theta_{j,}$$

gdje je θ_j kut između težina W_j i značajki x_i.

Sljedeći korak je L2-normalizacija težina W_j i značajki x_i: $\|W_j\| = 1 i \|x_i\| = 1$
i skaliranje rezultata sa s. Normalizacija rezultira time da predikcije ovise samo o kutu između težina i značajki. Naučeni ugrađeni vektori značajki su zbog toga distribuirani na hipersferi s radijusom s.
Normalizacija težina i značajki uklanja varijacije u radijusu značajki i osigurava to da su sve značajke distribuirane na hipersferi s radijusom s [23], kako je prikazano na primjeru na Slika 2.10.



Slika 2.10: Distribucija značajki na hipersferi uz radijus s=64

Nakon normalizacije težina i značajki formula L_1 prelazi u:

$$L_2 = \frac{-1}{N}\sum_{i=1}^{N} log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}}+\sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}}.$$

SphereFace rad [19] uvodi multiplikativnu marginu m i funkciju gubitka A-softmax koja je definirana sljedećom formulom:

$$L_3 = \frac{-1}{N}\sum_{i=1}^{N} log\frac{e^{s\,cos(\mathrm{m}\,\theta_{y_i})}}{e^{s\,cos}},$$

gdje je $\theta_{y_i} \in \left[0,\frac{\pi}{m}\right]$.

Da bi se uklonilo navedeno ograničenje na kut θ_{y_i}, izraz $cos\theta_{y_i}$ zamijenjen je monotonom funkcijom $\psi(\theta_{y_i})$, koja je definirana sa:

$\psi(\theta_{y_i}) = \frac{(-1)^k cos\,(\mathrm{m}\,\theta_{y_i})-2k+\lambda cos\theta_{y_i}}{1+\lambda}, \theta_{y_i} \in \left[\frac{k\pi}{m},\frac{(k+1)\pi}{m}\right], k \in [0, m-1], m \geq 1,$
gdje je λ dodatni hiperparametar zadužen za kontroliranje konvergencije učenja.

Time formula L_3 prelazi u:

$$L_4 = \frac{-1}{N}\sum_{i=1}^{N} log\frac{e^{s\,\psi(\theta_{y_i})}}{e^{s\psi(\theta_{y_i})}+\sum_{j=1,j\neq y_i}^{n} e^{s\,cos\theta_j}}.$$

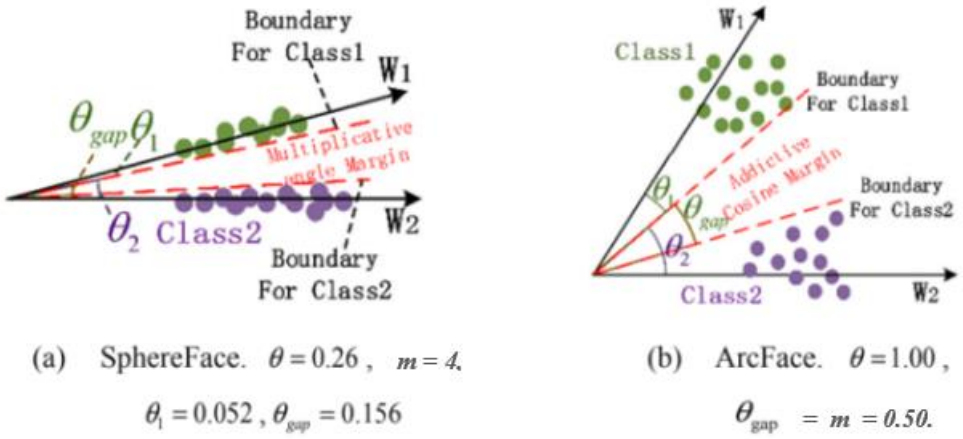
Klasifikacijska granica između dvije klase $C_1 i C_2$. definirana je kao:

$$\|x\|\big(cos(m\theta_1) - cos(\theta_2)\big) = 0,$$

gdje je θ_i maksimalni kut između centra klase $C_i, i = 1{,}2$ i primjera te klase. Neka je θ kut između centara klasa $C_1\, i\, C_2$. Onda vrijedi:

$$\theta_1 = \frac{1}{1+m}\theta.$$

Lijeva strana Slika 2.11 prikazuje geometrijsku interpretaciju A-softmax gubitka. Normalizirane težine W_1 i W_2 prestavljaju centre klasa $C_1\, i\, C_2$. Funkcija kutne multiplikativne konstante m je kompresija kutnog prostora koji pripada svakoj od klasa što dovodi do poboljšanja efikasnosti modela za raspoznavanje lica.



Slika 2.11: Geometrijska interpretacija A-softmax (SphereFace) gubitka i ArcFace gubitka[4]

Međutim, A-softmax ima i nekoliko mana. Zbog uvođenja hiperparametra λ modeli sa A-softmax gubitkom imali su problem nestabilnosti tokom učenja. Nadalje, kut θ_i ovisi o kutu θ, odnosno maksimalni kut između primjera neke klase i i pripadajućeg centra klase ovisi o kutu između centara različitih klasa. Budući da se centri dobiju normalizacijom težina zadnjeg sloja, a te težine su parametri koji se ugađaju tijekom učenja modela, može doći do toga da je konačni kut θ_i za neke klase veći, a za neke manji, što smanjuje diskriminativnu moć modela.

ArcFace gubitak temelji se na istim pretpostavkama kao i A-softmax gubitak, ali rješava spomenute probleme A-softmaxa. Izveden je, kao i A-softmax, iz softmax gubitka normalizacijom težina i značajki i skaliranjem rezultata sa s, čime se dobila formula:

$$L_2 = \frac{-1}{N}\sum_{i=1}^{N} log\frac{e^{s\,cos\theta_{y_i}}}{e^{s\,cos\theta_{y_i}} + \sum_{j=1,j\neq y_i}^{n} e^{s\,cos\theta_j}}.$$

ArcFace [20] uvodi aditivnu kutnu marginu m koja povećava unutarklasnu kompaktnost i međuklasnu različitost. ArcFace gubitak definiran je sljedećom jednadžbom:

$$L_5 = \frac{-1}{N}\sum_{i=1}^{N} log\frac{e^{s\,cos(\theta_{y_i}+m)}}{e^{s\,cos\left(\theta_{y_i}+m\right)} + \sum_{j=1,j\neq y_i}^{n} e^{s\,cos\theta_j}}.$$

[4] Slika preuzeta s adrese: https://cutt.ly/uuyIVTY

Klasifikacijska granica između dvije klase C_1 *i* C_2 definirana je kao:

$$\|x\|\big(cos(\theta_1 + m) - cos(\theta_2)\big) = 0,$$

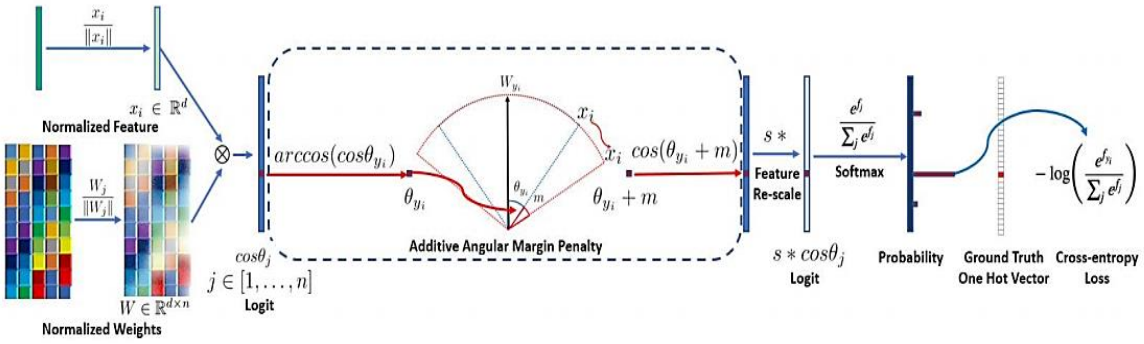
gdje je θ_i maksimalni kut između centra klase $C_i, i = 1,2$ i primjera te klase. Neka je θ kut između centara klasa C_1 *i* C_2.

Onda maksimalna udaljenost između primjera neke klase i pripadajućeg centra klase iznosi

$$\theta_i = \frac{\theta - m}{2}.$$

Možemo pisati da je $\theta_1 = \frac{\theta - m}{2}$ i $\theta_2 = \frac{\theta - m}{2}$. Odnosno, $\theta_1 + \theta_2 = \frac{\theta - m}{2} * 2 = \theta - m$ iz čega slijedi da je $\theta = \theta_1 + \theta_2 + m$. Dakle, aditivna kutna margina m definira minimalnu kutnu udaljenost između dva najsličnija primjera različitih klasa, što prikazuje Slika 2.11.

A-softmax gubitak ne definira eksplicitno minimalnu udaljenost između dva najsličnija primjera različitih klasa. Nadalje, ta udaljenost je manja u slučaju korištenja A-softmax gubitka u odnosu na ArcFace gubitak, zbog čega modeli sa A-softmax gubitkom imaju veću vjerojatnost pogrešne klasifikacije sličnih primjera različitih klasa.



Slika 2.12: Računanje izlaza modela s ArcFace gubitkom[5]

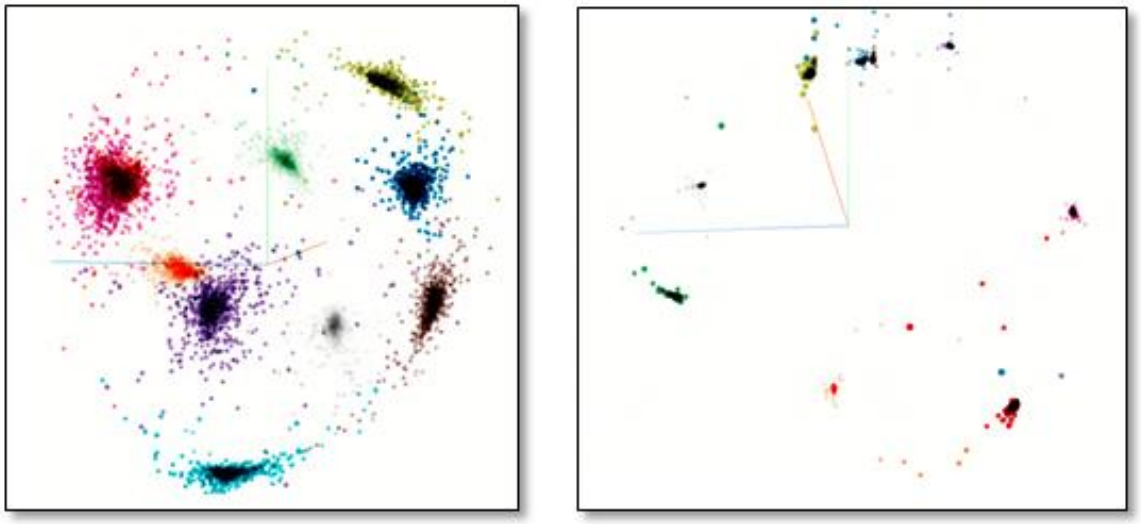
Uvijet za ekifasan model za raspoznavanje lica s otvorenim skupom jest da ugrađeni vektor značajki dobiveni modelom zadovoljavaju kriterij da je maksimalna udaljenost unutar klase manja od minimalne udaljenosti među klasama pod određenim metričkim prostorom.

[5] Slika preuzeta iz [20]

Slika 2.13 prikazuje vektore značajki dobivene VGG16 modelom učenim na MNIST skupu podataka korištenjem softmax gubitka i ArcFace gubitka. MNIST skup podataka je skup rukom pisanih znamenki, što znači da sadrži 10 klasa.

Vidljivo je softmax gubitak ne osigurava dovoljno veliku unutarklasnu kompaktnost, kao ni veliku međuklasnu udaljenost naučenih značajki, posebno ako se uzme u obzir da je broj klasa dosta malen (10).

S druge strane, značajke dobivene modelom sa ArcFace gubitkom zadovoljavaju uvjet unutarklasne kompaktnosti i velike međuklasne udaljenosti naučenih značajki.
Zbog toga je ArcFace gubitak pogodniji za problem raspoznavanja lica, budući da skupovi podataka za raspoznavanje lica često sadrže tisuće klasa s velikim varijacija unutar jedne klase i velikim stupnjem sličnosti primjera različitih klasa.

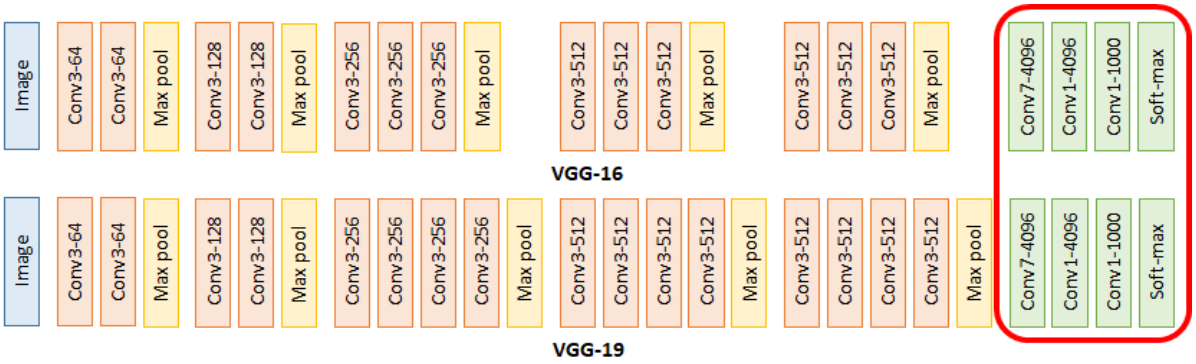


Slika 2.13: Vektori značajki dobiveni VGG16 modelom učenim na MNIST skupu podataka korištenjem softmax gubitka (lijevo) i ArcFace gubitka (desno)[6]

[6] Slike preuzete s adrese: https://cutt.ly/FuqPRlU

2.4 Moderne arhitekture konvolucijskih neuronskih mreža

Prve arhitekture konvolucijskih mreža najčešće su građene uzastopnim nizanjem konvolucijskih slojeva i slojeva sažimanja nakon čega bi uslijedilo nekoliko potpuno povezanih slojeva prije izlaza iz mreže. Izlaz iz jednog sloja bio je ulaz u sljedeći sloj. Poznata arhitektura konvolucijske mreže koja slijedi taj obrazac jest VGGNet [4] koja je predstavljena 2014. u sklopu ILSCVR natjecanja (engl. *ImageNet Large Scale Recognition Challenge*). Arhitektura se pokazala jako uspješna u savladavanju problema lokalizacije i klasifikacije objekata.

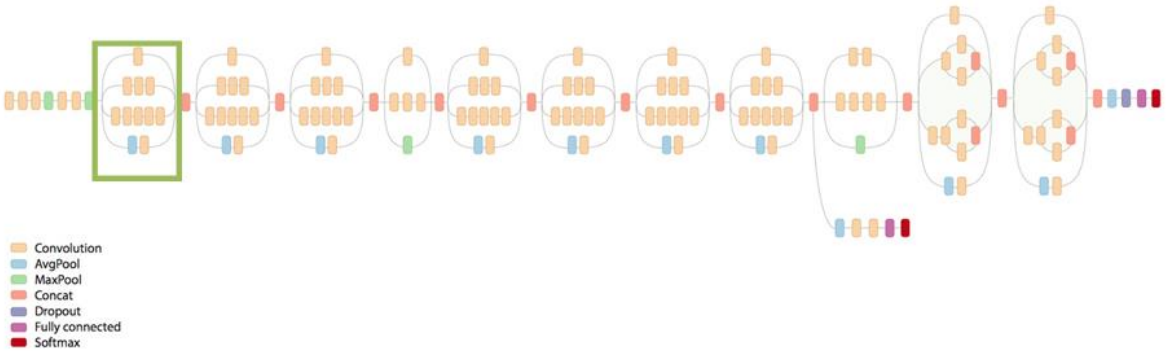


Slika 2.14: Arhitektura mreža VGG-16 i VGG-19[7]

Slika 2.14 prikazuje arhitekture mreža VGG-16 (koja ima 16 slojeva s težinama) i VGG-19 (koja ima 19 slojeva s težinama).
Iako je imala jako dobre rezultate, VGGNet je osvojila tek drugo mjesto na ILSCVR2014 natjecanju. Prvo mjesto osvojio je tim iz Googlea sa mrežom GoogLeNet.

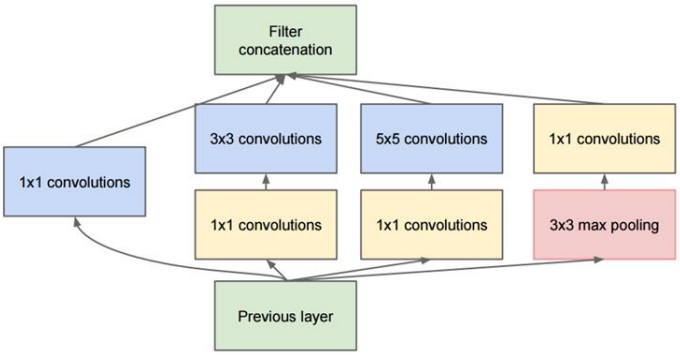
Arhitektura mreže GoogLeNet potpuna je suprotnost jednostavnoj i klasičnoj arhitekturi mreža VGGNet. Naime, te godine tim iz Googlea koji je sudjelovao na ILSCVR natjecanju, predstavio je takozvani „Inception blok" (engl. *Inception module*) i time prekinuo do tada ustaljeni pristup izrade dubokih mreža slaganjem konvolucijskih slojeva i slojeva sažimanja u sekvencijalnu strukturu.

[7] Slika preuzeta s adrese: https://bit.ly/2U8Ooh3



Slika 2.15: Arhitektura mreže GoogLeNet[8]

Slika 2.15 prikazuje arhitekturu GoogLeNet mreže. Odmah na prvi pogled vidljivo je da u mreži postoje dijelovi koji se izvode paralelno. Dio označen zelenim kvadratom je jedan Inception blok. Njegova struktura prikazana je na sljedećoj slici.



Slika 2.16: Inception blok iz GoogLeNet[9]

Inception blok može se shvatiti kao mreža u mreži. Predstavljanjem ovakvih modula, autori GoogLeNet [5] mreže predstavili su novi pristup izgradnji dubokih modela i pokazali da kreativna struktura slojeva, drukčija od one klasične sekvencijalne, može dovesti do poboljšanja performansi modela [9].

Time je počeo trend izgradnje novih i efikasnijih arhitektura dubokih mreža, poput ResNet [6], Xception [7] ili MobileNet-V1 [8]. ResNet i varijanta Xception arhitekture korištene su u programskom dijelu ovog rada i opisane detaljnije u nastavku.

[8] Slika preuzeta s adrese: https://bit.ly/2BmKScj

[9] Slika preuzeta s adrese: https://bit.ly/2BmKScj

2.4.1 1x1 konvolucija

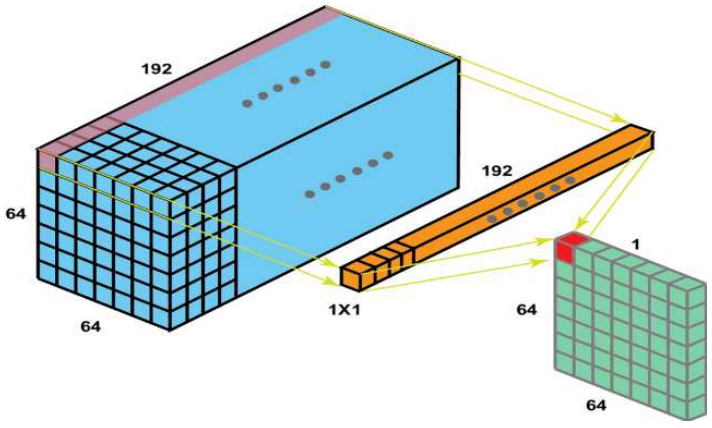
1x1 konvolucijski sloj predstavljen je prvi puta u radu „*Network in network*“ [22] autora Min Lin et all 2013. godine. Korišten je za smanjenje dimenzionalnosti duž osi dubine (engl. *cross channel down sampling*).

1x1 konvolucija zapravo je klasična operacija konvolucije uz korištenje filtera veličine *1 x 1 x dubina_ulaza*.

Ako imamo ulazni volumen F dimenzija $D_F x D_F x M$, gdje je D_F širina i visina ulaznog volumena, a M dubina ulaznog volumena (broj kanala). Ako je ulaz RGB slika, onda je $M = 3$.

Kada nad ulazom F obavimo operaciju konvolucije koristeći jedan filter dimenzija $1x1xM$, dobijemo izlaz dimenzija $D_F x D_F x 1$. Ako primijenimo N takvih filtera, konačna dimenzija izlaza bit će $D_F x D_F x N$.

U situacijama kada je M jako velik, kao na Slika 2.17, gdje je $M = 192$, primjena N filtera dimenzija $1x1xM$, gdje je $N < M$, dovodi do smanjenja dimenzionalnosti izlaza (engl. *dimensionality reduction*) duž dimenzije dubine, **uz očuvanje dimenzija širine i visine (receptivnog polja)** ulazne mape značajki.



Slika 2.17: Rezultat 1x1 konvolucije nad ulaznim volumenom dimenzija 64x64x192 koristeći jedan filter dimenzija 1x1x192[10]

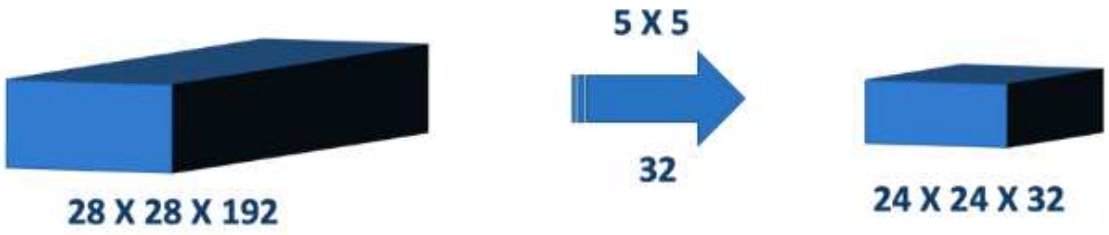
[10] Slika preuzeta s adrese: https://cutt.ly/vy0pj1E

1x1 konvolucija u praksi ima nekoliko primjena. To su:

a) modifikacija dimenzionalnosti duž dimenzije dubine, kako je opisano gore
b) smanjenje računske složenosti
c) dodavanje nelinearnosti u mežu
d) stvaranje dubljih modela pomoću slojeva uskog grla (engl. *bottle-neck*), kao u ResNet mrežama
e) stvaranje konvolucijskih mreža s manjim brojem parametara koje imaju bolje performanse od arhitektura koje koriste samo klasične konvolucijske slojeve.

Modifikacija dimenzionalnosti podrazumijeva promjenu dimenzije dubine, uz očuvanje dimenzija širine i visine (receptivnog polja) ulazne mape značajki. Teoretski, ta promjena može značiti i povećanje i smanjenje broja kanala jer broj kanala izlaza odgovara broju 1x1 filtera koji se koriste. Međutim, 1x1 konvolucija najčešće se koristi za redukciju (smanjenje) broja kanala prije računski skupih operacija poput 3x3 ili 5x5 konvolucije [5], jer se time postiže i smanjenje računske složenosti.

Promotrimo klasičnu operaciju konvolucije nad ulazom dimenzija *28x28x192* uz korištenje 32 filtera dimenzija *5x5* (*5x5x192*), prikazanu na Slika 2.18.



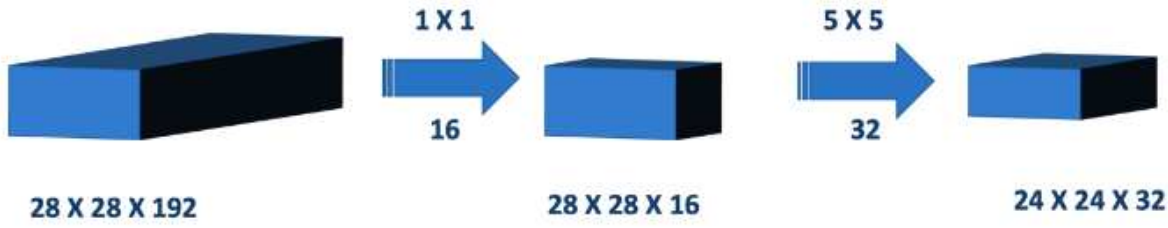
Slika 2.18: Rezultat klasične operacije konvolucije nad ulazom dimenzija 28x28x192 uz korištenje 32 filtera dimenzija 5x5 (5x5x192)

Broj operacija množenja potrebnih za obavljanje te operacije iznosi
*(5*5*192)*(24*24)*32 = 88 473 600.*
Dodavanjem 1x1 konvolucijskog sloja prije klasičnog konvolucijskog sloja koji koristi 5x5 filtere, zadržavamo dimenzije širine i visine prvotne ulazne mape značajki, ali smanjuje se broj potrebnih operacija množenja te time povećava učinkovitost modela.

Na Slika 2.19 prikazana je operacija konvolucije sa Slika 2.18 uz dodavanje 16 filtera dimenzija 1x1x192.

Broj operacija množenja potrebnih za obavljanje 1x1 konvolucija i dobivanje izlaznog volumena dimenzija *28x28x16* iznosi *(1*1*192)*(28*28)*16 = 2 408 448.*
Broj operacija množenja potrebnih za obavljanje 5x5 konvolucija i dobivanje konačnog izlaza dimenzija *24x24x32* iznosi *(5*5*16)*(24*24)*32 = 7 372 800.*
Zajedno, potrebno je *9 781 248* operacija množenja.

U oba slučaja prikazana (Slika 2.18, Slika 2.19) obavlja se konvolucija nad ulazom dimenzija *28x28x192* i dobije izlaz dimenzija *24x24x32.*
Međutim, uz korištenje 1x1 konvolucije, broj potrebnih operacija množenja smanjuje se $\frac{88473600}{9781248} \approx 9$ puta.



Slika 2.19: Rezultat operacije konvolucije nad ulazom dimenzija 28x28x192 uz korištenje 32 filtera dimenzija 5x5 (5x5x192) uz dodavanje 1x1 konvolucije

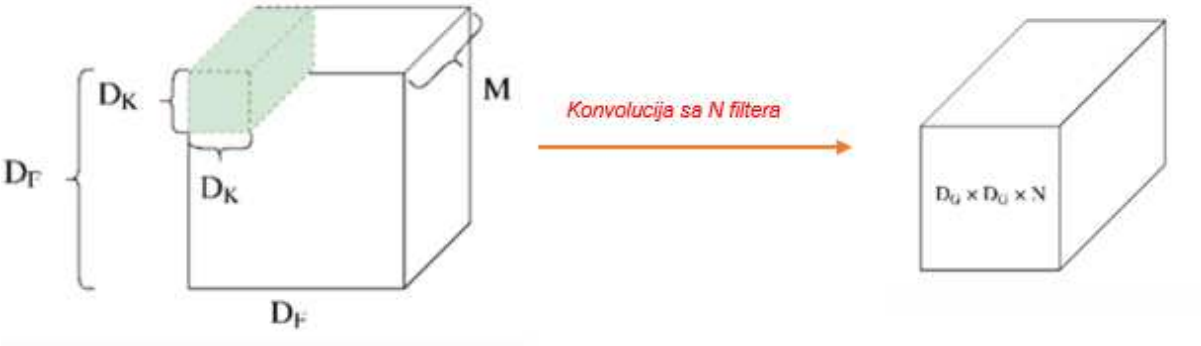
Ova primjena 1x1 konvolucijskih slojeva može se vidjeti i na Slika 2.16 u Inception bloku GoogLeNet mreže.

Klasični konvolucijski slojevi računaju sumu produkata elemenata filtera s odgovarajućim elementima receptivnog polja i rezultate prosljeđuju aktivacijskoj funkciji. Rezultat koji se dobije je mapa značajki.
Uvođenjem dodatnog sloja 1x1 konvolucije u mrežu, uvodi se i dodatni sloj nelinearne aktivacije. Dakle, 1x1 konvolucija koristi se za smanjenje broja kanala ulaza (i smanjenje računske složenosti) uz uvođenje dodatne nelinearnosti u mrežu.

2.4.2 Dubinski odvojiva konvolucija

Izvođenje standardne operacije konvolucije opisane u poglavlju 2.1 vremenski je dosta zahtjevno. Kao rješenje tog problema predložena je alternativna metoda izvođenja operacije konvolucije u dubokim neuronskim mrežama, a naziva se dubinski odvojiva (engl. *depthwise separable*) konvolucija.

Ako imamo ulazni volumen F denzija $D_F x D_F x M$, gdje je D_F širina i visina ulaznog volumena, a M dubina ulaznog volumena (broj kanala). Ako je ulaz RGB slika, onda je $M = 3$.
Kada nad ulazom F obavimo operaciju konvolucije koristeći filter dimenzija $D_K x D_K x M$, dobijemo izlaz dimenzija $D_G x D_G x 1$. Ako primijenimo N takvih filtera, konačna dimenzija izlaza bit će $D_G x D_G x N$.



Slika 2.20: Konvolucija ulaznog volumena dimenzija D_FxD_FxM s N filtera dimenzija D_KxD_KxM

Operacija konvolucije računa sumu produkata elemenata ulaza s filterima i vraća skalar. Filter se pomiče po ulaznom volumenu i računanje sume produkata obavlja se nakon svakog pomaka.
Budući da je operacija množenja skuplja od operacije zbrajanja, kompleksnost izvođenja operacije konvolucije možemo izraziti u broju operacija množenja koje je potrebno obaviti tokom jedne operacije konvolucije. Za jednu operaciju konvolucije broj operacija množenja koji se obavi jednak je broju elemenata u filteru, jer se svaki od tih

elemenata pomnoži s odgovarajućim elementima ulaznog volumena. Dakle, za **jednu operaciju konvolucije** obavi se $D_K * D_K * M = {D_K}^2 * M$ operacija množenja.

Međutim, filter se pomiče po ulaznom volumenu duž dimenzija širine i visine te nakon svakog pomaka obavi se operacija konvolucije. Ukupan broj pomaka, odnosno **ukupan broj operacija konvolucije**, koje se **jednim filterom** obave nad cjelokupnim ulaznom volumenom iznosi $D_G * D_G = {D_G}^2$.
Znači da je ukupan broj operacija množenja potreban za operaciju konvolucije jednim filterom nad cjelokupnim ulaznim volumenom jednak ${D_G}^2 * {D_K}^2 * M$.

Na kraju, budući da imamo N filtera, ukupan broj operacija množenja potreban da se obavi konvolucija ulaznog volumena dimenzija $D_F x D_F x M$ s N filtera dimenzija $D_K x D_K x M$ iznosi $N * {D_G}^2 * {D_K}^2 * M$.

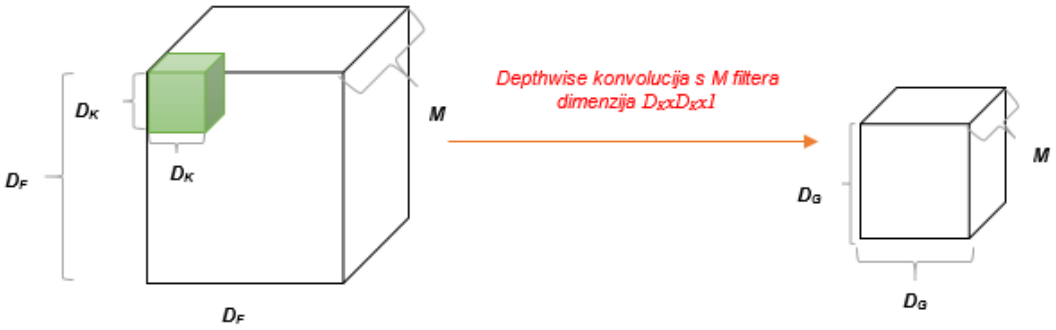
Pri izvođenju klasične operacije konvolucije aplikacija filtera nad svim kanalima ulaznog volumena i spajanje svih tih vrijednosti obavlja se u jednom koraku.
Dubinski odvojiva konvolucija obavlja isto, ali u 2 koraka:

1. Prvi korak je faza filtriranja ili dubinska konvolucija (engl. *Depthwise convolution – filtering stage*).
2. Drugi korak je faza spajanja ili 1x1 konvolucija (engl. *Pointwise convolution – combining stage*).

Prvi korak – dubinska konvolucija

U prvom koraku obavlja se konvolucija, ali ne nad svim kanalima dubine odjednom kao kod klasične konvolucije, već nad jednim po jednim kanalom dubine.
Uzmimo za primjer opet onaj isti ulazni volumen F dimenzija $D_F x D_F x M$, gdje je D_F širina i visina ulaznog volumena, a M dubina ulaznog volumena (broj kanala).
U prvom koraku dubinski odvojive konvolucije koriste se filteri dimenzija $D_K x D_K x 1$, a ne $D_K x D_K x M$ kao u klasičnoj konvoluciji.

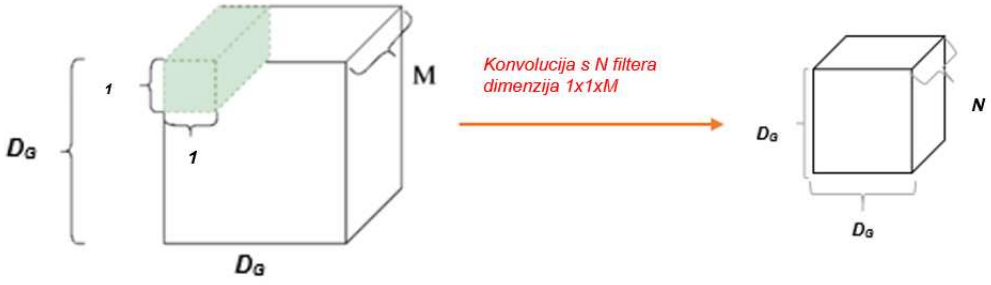
Svakim od tih filtera obavlja se konvolucija s 2D dijelom ulaznog volumena na određenoj dubini. Budući da imamo M takvih dijelova, po jedan za svaki od M kanala dubine ulaznog volumena, potrebno nam je ukupno M filtera dimenzija $D_K x D_K x 1$.
Primjena jednog ovakvog filtera nad opisanim dijelom ulaznog volumena proizvodi izlaz dimenzija $D_G x D_G x 1$.
Primjenom M takvih filtera na cijeli ulazni volumen daje izlaz dimenzija $D_G x D_G x M$, što je ujedno i rezultat prvog koraka dubinski odvojive konvolucije, kako prikazuje Slika 2.21.



Slika 2.21: Depthwise separable konvolucija - prvi korak

Drugi korak – 1x1 konvolucija

U drugom koraku dubinski odvojive konvolucije ulazni volumen je rezultat prvog koraka dimenzija $D_G x D_G x M$. Filteri u ovom koraku imaju dimenziju $1x1xM$. Dakle, obavlja se 1x1 konvolucija nad izlazom prvog koraka.
Rezultat operacije konvolucije jednim takvim filterom nad ulaznim volumenom ima dimenzije $D_G x D_G x 1$.
Neka je ukupan broj 1x1 filtera koji se koriste u ovom koraku jednak N. Onda je dimenzija konačnog izlaza nakon drugog koraka dubinski odvojive konvolucije jednaka dimenziji izlaza klasične konvolucije i iznosi $D_G x D_G x N$.
Međutim, ovakav način obavljanja konvolucije je računski isplativiji.



Slika 2.22: Depthwise separable konvolucija - drugi korak

Prvi korak dubinski odvojive konvolucije zahtjeva $D_G{}^2 * D_K{}^2 * M$ operacija množenja:

- U svakom koraku pomicanja filtera dimenzija $D_K x D_K x 1$ po ulaznom volumenu obavi se $D_K * D_K = D_K{}^2$ operacija množenja.
- Svaki filter se pomiče po ulaznom volumenu i generira izlaz dimenzija $D_G x D_G x 1$ što znači da svaki filter $D_G{}^2$ puta obavi $D_K{}^2$ operacija množenja.
- Ako koristimo M takvih filtera, konačan broj operacija množenja u prvom koraku iznosi $D_G{}^2 * D_K{}^2 * M$.

Drugi korak dubinski odvojive konvolucije zahtjeva $N * D_G{}^2 * M$ operacija množenja:

- U svakom koraku pomicanja filtera dimenzija $1x1xM$ po ulaznom volumenu obavi se M operacija množenja.
- Svaki filter se pomiče po ulaznom volumenu dimenzija $D_G x D_G x M$ što znači da svaki filter $D_G{}^2$ puta obavi M operacija množenja.
- Ako koristimo N takvih filtera, konačan broj operacija množenja u drugom koraku iznosi $N * D_G{}^2 * M$.

Ukupan broj operacija množenja u oba koraka dubinski odvojive konvolucije iznosi $D_G{}^2 * D_K{}^2 * M + N * D_G{}^2 * M = M * D_G{}^2 (D_K{}^2 + N)$.

Označimo sa *A* broj operacija množenja potrebnih za obavljanje opisane klasične konvolucije, a s *B* broj operacija množenja potrebnih za obavljanje dubinski odvojive konvolucije.

$$\frac{B}{A} = \frac{M * D_G{}^2 * (D_K{}^2 + N)}{N * D_G{}^2 * D_K{}^2 * M} = \frac{1}{N} + \frac{1}{D_K{}^2}.$$

Ako uzmemo kao primjer da je $N = 1024$ i $D_K = 3$, dobijemo da je

$$\frac{B}{A} = \frac{1}{1024} + \frac{1}{3^2} = 0.112.$$

Vidljivo je da je dubinski odvojiva konvolucija u odnosu na klasičnu operaciju konvolucije računski znatno manje zahtjevna.

Dubinski odvojiva konvolucija također zahtjeva i manje parametara od klasične konvolucije.

Za klasičnu konvoluciju (Slika 2.20) korišteno je N filtera od kojih svaki ima dimenzije $D_K x D_K x M$ dakle ukupan broj parametara iznosi ${D_K}^2 * M * N.$

Za dubinski odvojivu konvoluciju (Slika 2.21, Slika 2.22) korišteno je M filtera dimenzija $D_K x D_K x 1$ a zatim N filtera dimenzija $1x1xM$. Ukupno to je ${D_K}^2 * M + N * M,$ odnosno $M * ({D_K}^2 + N)$ parametara.

Omjer je jednak kao i omjer broja operacija množenja i iznosi:

$$\frac{M * \left({D_K}^2 + N\right)}{{D_K}^2 * M * N} = \frac{1}{N} + \frac{1}{{D_K}^2}.$$

Modeli koji koriste dubinski odvojivu konvoluciju manji su od modela koji koriste klasične konvolucijske slojeve (imaju manje parametara), računski su efikasniji i imaju bolje performanse.

Zbog navedenih prednosti u odnosu na standardnu operaciju konvolucije, dubinski odvojiva konvolucija koristi se u mnogim modernim arhitekturama konvolucijskih mreža, poput Xception arhitekture i MobileNet-V1 arhitektura.

2.4.3 ResNet arhitektura

ResNet (engl. *Residual Network*) je vrsta neuronske mreže koja se pokazala jako uspješna u rješavanju zadataka iz domene računalnog vida. Predstavljena je 2015. godine u sklopu ILSVRC natjecanja i smatra se jednim od najznačajnijih postignuća u području dubokog učenja, jer rješava problem iščezavajućih gradijenata (engl. *vanishing gradients*) i omogućava učenje jako dubokih modela (s preko 150 slojeva).

Prije ResNet arhitekture učenje dubokih modela algoritmom širenja greške unatrag, poznatijeg kao backpropagation, nije bilo tako jednostavno. Backpropagation je jednostavan i računski nezahtjevan način računanja gradijenta kompozicije funkcija. Ideja je određivanje greške modela i gradijenata u svakom sloju te ažuriranje težina na temelju gradijenata (gradijentni spust) [16].

Učenje neuronske mreže provodi se u dva koraka. U prvom koraku dovodi se poznati uzorak na ulaz, računa se odziv mreže te se računa pogreška (razlika odziva mreže i očekivanog izlaza za dati poznati uzorak). Dakle, najprije se izračunava greška izlaznog sloja. Zatim se za prethodni sloj određuje koliko je svaki neuron utjecao na greške u idućem sloju te se računaju greške tih neurona. Zatim se određuje gradijent greške po pojedinim težinama koju povezuju te slojeve te se one ažuriraju.
Na taj način greška se širi unatrag i obavlja se ažuriranje parametara kako bi se u idućem koraku greška smanjila. Ideja je iterativno optimizirati parametre u mreži s obzirom na funkciju gubitka.

Deriviranje kompozicije funkcija obavlja se pomoću pravila ulančavanja (engl. *chain rule*). Za vektorske funkcije $z = f(x, y), x = g(t), y = h(t)$, gradijent od z s obzirom na t računa se kao:

$$\frac{\partial z}{\partial t} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t}.$$

Promotrimo kao primjer duboki model f parametriziran s θ koji podatke x preslikava u predikcije $\hat{y}$:

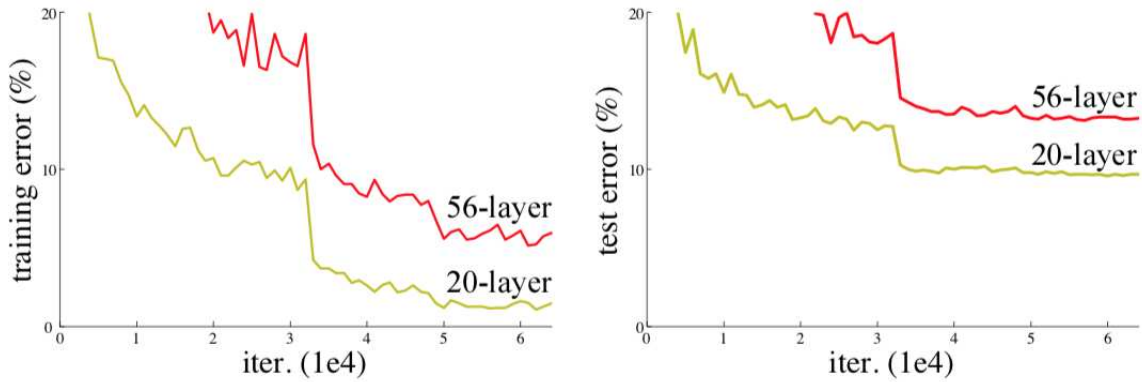
$$\hat{y} = f(\theta, x).$$

Neka je y očekivani izlaz mreže, $L(y, \hat{y})$ gubitak (greška) mreže za ulaz x, $a^{[l]}$ izlaz iz sloja l i L ukupan broj slojeva u mreži. Gradijent gubitka s obzirom na parametre sloja l tada je:

$$\frac{\partial L(y,\hat{y})}{\partial \theta^{[l]}} = \frac{\partial L(y,\hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a^{[L]}} \frac{\partial a^{[L]}}{\partial a^{[L-1]}} \frac{\partial a^{[L-1]}}{\partial a^{[L-2]}} \cdots \frac{\partial a^{[l]}}{\partial \theta^{[l]}}.$$

Da bi se gradijent gubitka propagirao kroz mrežu od kraja mreže prema početku, koristi se pravilo ulančavanja koje množi parcijalnu derivaciju sloja l s parcijalnim derivacijama slojeva od $(l+1)$ do L. Budući da te parcijalne derivacije mogu biti jako male (iz intervala [0,1]), propagacijom gradijenta prema početnim slojevima gradijent postaje sve manji i manji te polako iščezava, što znači da težine i pragovi početnih slojeva mreže neće biti dovoljno efikasno ažurirani tokom učenja.

Iako bi u teoriji dublje mreže trebale imati bolje performanse, zbog problema iščezavajućih gradijenata klasične arhitekture mreža (engl. *plain network,* primjer: VGGNet) bilježe porast greške na skupu za učenje s povećanjem broja slojeva, odnosno korištenje dubljih modela degradira performanse modela, kako prikazuje Slika 2.23.

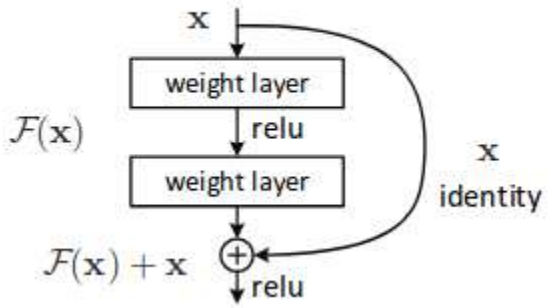


Slika 2.23: Greška učenja (lijevo) i greška testiranja (desno) na skupu CIFAR-10 korištenjem klasičnih modela sa 20 slojeva i 56 slojeva[11]

[11] Slika preuzeta iz [6]

Uzmimo za primjer dva modela, jedan manji (s manjim brojem slojeva) i drugi dublji, koji je izgrađen tako da su na slojeve prvog modela dodani slojevi koji obavljaju funkciju identiteta (izlaz iz sloja jednak je ulazu u sloj). U ovakvoj situaciji dublji model trebao bi imati grešku na skupu za učenje barem jednaku onoj koju ima manji model, ako ne i manju.
Međutim, eksperimenti su pokazali upravo suprotno – korištenje dubljeg modela degradira performanse, što sugerira da višestruki nelinerani slojevi imaju poteškoća s aproksimacijom funkcije identiteta [6].

Neka je $H(x)$ osnovna funkcija koju aproksimira skup od nekoliko uzastopnih slojeva mreže (ne mora biti cijela mreža), od kojih prvi ima ulaz označen sa x. Autori [6] uvode takozvane preskočne veze (engl. *skip connections*) koje omogućavaju da se aktivacija iz jednog sloja l_i dovede na ulaz dubljeg sloja l_j tako da vrijedi da je $j > i + 1$. Preskočne veze još se nazivaju i prečaci (engl. *shortcuts*).



Slika 2.24: Rezidualni blok (engl. residual block)[12]

$\mathrm{F}(x)$ na Slika 2.24 predstavlja rezidualnu funkciju koja je definirana kao razlika između izlaza i ulaza skupa uzastopnih slojeva, odnosno $\mathrm{F}(x) = H(x) - x$. Originalna osnovna funkcija koju aproksimira skup od nekoliko uzastopnih slojeva mreže tada izgleda ovako: $H(x) = \mathrm{F}(x) + x$. Ideja rezidualnog bloka proizišla je iz pretpostavke da je jednostavnije aproksimirati rezidualnu funkciju nego originalnu funkciju. Nadalje, ako je ciljana funkcija koju je potrebno aproksimirati zapravo funkcija identiteta $H(x) = x$,

[12] Slika preuzeta iz [6]

jednostavnije je naučiti rezidualnu funkciju da ima vrijednost nula, nego naučiti funkciju identiteta originalnim skupom nelinearnih slojeva [6].

Preskočne veze imaju sljedeće funkcije unutar mreže :

- Pružaju način prenošenja informacije iz ranijih slojeva u mreži u one kasnije.
- Rješavaju problem iščezavajućih gradijenata, jer pruža alternativni put kroz mrežu kojim se gradijent može propagirati prema početnim slojevima.
- Omogućavaju modelu da nauči funkciju identiteta.

Rezidualni blok sa Slika 2.24 obavlja preslikavanje:

$$y = ReLU\ (\ \mathrm{F}(x, \{W_i\}) + x\ ) = ReLU\ (\ [\ W_2\ ReLU(W_1 x + \ b_1) + b_2] + x),$$

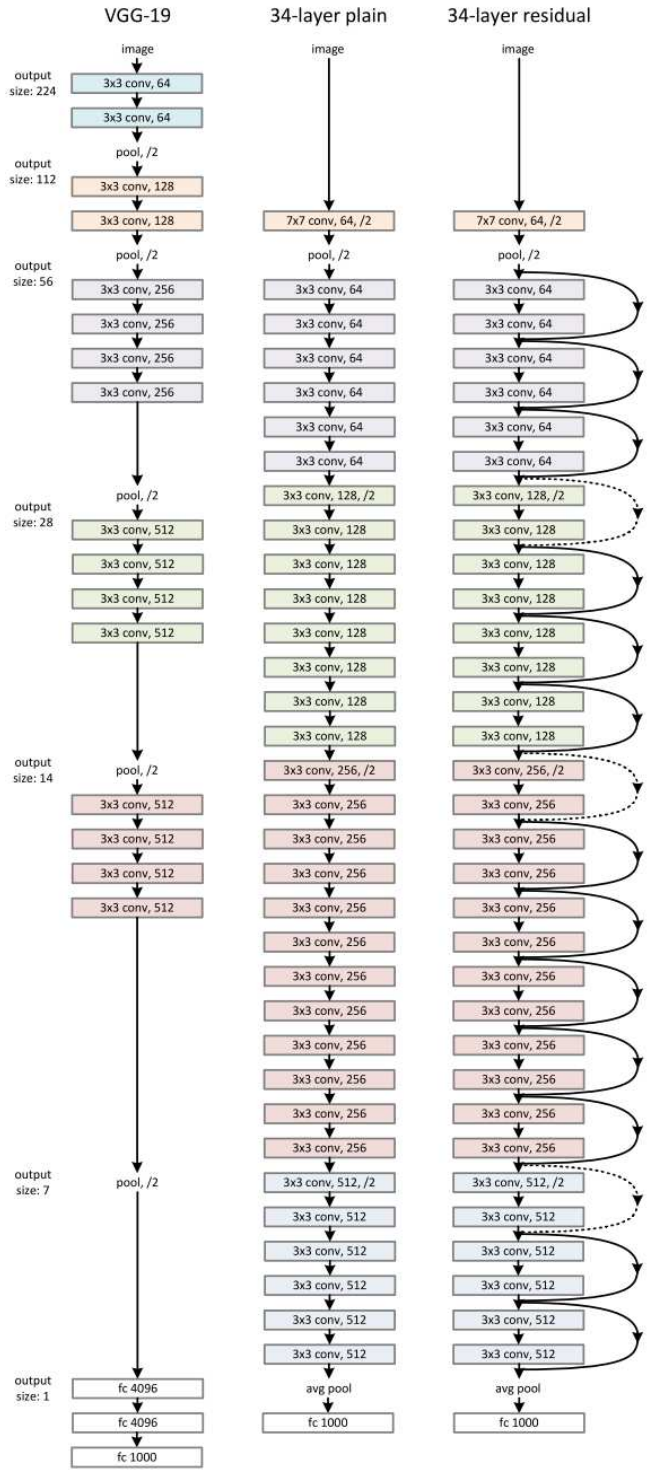
gdje su x i y ulaz i izlaz skupa slojeva sa slike, (W_1, b_1) i (W_2, b_2) su težine i pragovi slojeva. Preskočna veza sa slike označena sa „*x identity*“ ne uvodi dodatne parametre niti računsku složenost u mrežu, već samo prosljeđuje ulaz x dalje kroz mrežu gdje se on zbraja s izlazom y drugog sloja u bloku te se zbroj dovodi na ulaz nove $ReLU$ funkcije. Ovakva preskočna veza može se koristiti ukoliko su x i $\mathrm{F}(x, \{W_i\})$ jednakih dimenzija i naziva se identitet-prečac (engl. *identity shortcut*).
Ukoliko su x i $\mathrm{F}(x, \{W_i\})$ različitih dimenzija, izlaz skupa slojeva sa Slika 2.24 računa se kao :

$$y = ReLU\ (\ \mathrm{F}(x, \{W_i\}) + W_s x\ ) = ReLU\ (\ [\ W_2\ ReLU(W_1 x + \ b_1) + b_2] + W_s x),$$

gdje W_s obavlja linearnu projekciju u svrhu izjednačavanja dimenzionalnosti od x s dimenzionalnošću od $\mathrm{F}(x, \{W_i\})$ (u praksi se najčešće koristi 1x1 konvolucija). Preskočne veze koje obavljaju linearnu projekciju u svrhu izjednačavanja dimenzija od x i $\mathrm{F}(x, \{W_i\})$ nazivaju se projekcijski prečaci (engl. *projection shortcut*).

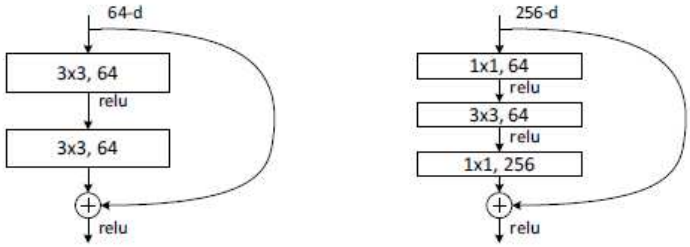
Slika 2.25 prikazuje arhitekture VGG-19, klasične (engl. *plain*) mreže sa 34 sloja i rezidualne mreže sa 34 sloja. Konvolucijski slojevi koji sačinjavaju rezidualne blokove koriste *3x3* filtere. Konvolucijski slojevi čije izlazne mape značajki imaju iste dimenzije širine i visine imaju jednak broj filtera. Smanjenje dimenzionalnosti ulaza ne obavlja se slojevima sažimanja, već konvolucijskim slojevima s korakom filtera 2. U slučaju smanjenja dimenzija širine i visine izlazne mape značajki dva puta, broj filtera se

poveća 2 puta. Pune strelice na slici označavaju identitet-prečace, a isprekidane strelice projekcijske prečace.



Slika 2.25: : VGG-19 mreža, klasična mreža sa 34 sloja i rezidualna mreža sa 34 sloja

Za izgradnju još dubljih mreža, [6] predlaže alternativni izgled rezidualnog bloka koji nazivaju konvolucijski blok s uskim grlom (engl. *bottleneck block*) u svrhu poboljšanja efikasnosti i smanjenja računske složenosti. Dva konvolucijska sloja s *3x3* filterima iz klasičnog rezidualnog bloka zamijenjena su s tri konvolucijska sloja s filterima veličine 1x1, 3x3, 1x1. Prvi sloj obavlja redukciju dimenzionalnosti 1x1 konvolucijom u svrhu smanjenja složenosti operacija u sljedećem sloju sa 3x3 filterima. Treći sloj koristi 1x1 konvoluciju za obnavljanje dimenzionalnosti izlaza.



Slika 2.26: Lijevo: klasični rezidualni blok. Desno: bottleneck blok[13]

Zamjenom klasičnih rezidualnih blokova iz ResNet-34 arhitekture, koju prikazuje Slika 2.25, s konvolucijski blokovima s uskim grlom dobije se ResNet-50 arhitektura. Projekcijski prečaci u ResNet-50 mreži obavljaju linearnu projekciju pomoću 1x1 konvolucije. Struktura mreže ResNet-50, kao i ostalih varijanti ResNet arhitekture predstavljenih u [6], prikazana je na slici ispod.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

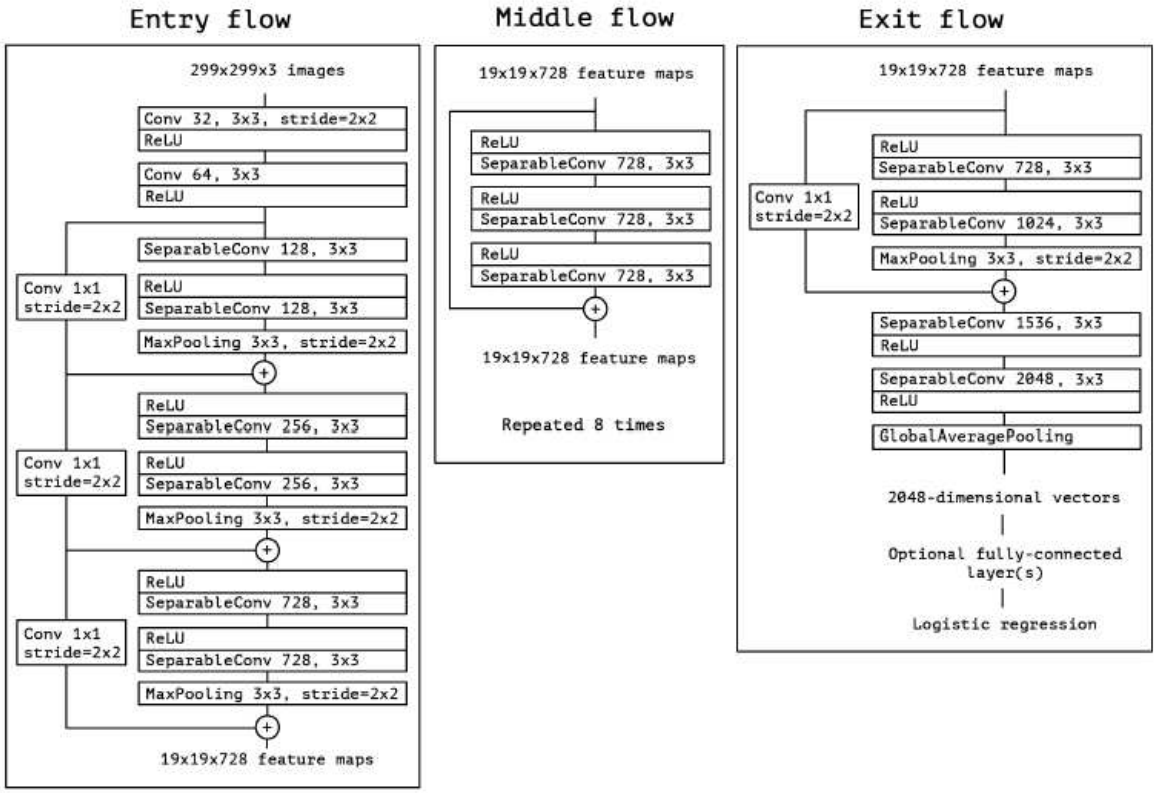
Slika 2.27: Različite ResNet arhitekture predstavljene u [6]

[13] Slika preuzeta iz [6]

2.4.4 Xception arhitektura

Xception arhitektura, predstavljena u [7], arhitektura je konvolucijskih neuronskih mreža koja se u potpunosti temelji na dubinski odvojivoj konvoluciji. Temeljna pretpostavka je da se mapiranja duž kanala dubine te mapiranja duž kanala širine i visine mogu u potpunosti odvojiti. Budući da je ova pretpostavka slična pretpostavci na kojoj se temelje Inception mreže, ova arhitektura nazvana je Xception što je skraćenica za „*Extreme Inception*“.

Ukratko, Xception arhitektura sastoji se od modificiranih slojeva dubinski odvojive konvolucije sa preskočnim vezama. Varijanta dubinski odvojive konvolucije korištena u ovoj arhitekturi prvo obavlja 1x1 konvoluciju, zatim prostornu konvoluciju po kanalima. Također, između ta dva koraka ne koristi se aktivacijska funkcija.



Slika 2.28: Xception arhitektura opisana u [7]

Slika 2.28 prikazuje arhitekturu Xception mreže. Nakon svakog konvolucijskog sloja, kao i sloja dubinski odvojive konvolucije, slijedi sloj normalizacije nad grupom (engl. *batch normalization*).

3. Detekcija napada u sustavima za autentikaciju analizom izgleda lica

Autentikacija korisnika je primarni sigurnosni mehanizam. Korisničke lozinke još uvijek su najšire korištena metoda autentikacije. Nedostatci autentikacije pomoću lozinke proizlaze iz činjenice da je snažne lozinke ljudima teško zapamtiti, dok one lakše pamtljive ne pružaju dovoljno snažnu zaštitu i najčešće se lako probiju.

Biometrijske metode autentikacije nameću se kao obećavajuća alternativa lozinkama i autentikaciji pomoću tokena.
Biometrija se bavi identifikacijom individualca na temelju nekih osobi jedinstvenih bioloških ili bihevioralnih karakteristika [1], poput otiska prsta, geometrije lica ili izgleda šarenice oka.
Autentikacija osobe analizom izgleda lica smatra se jednom od najmanje intruzivnih biometrijskih metoda [2] jer, barem u teoriji, osoba mora samo približiti lice kameri i sustav na temelju slike lica dobivene kamerom obavi autentikaciju.
Međutim, za razliku od drugih biometrijskih podataka poput otiska prsta ili izgleda šarenice oka koje je teško duplicirati, izgled lica osobe relativno je jednostavno zabilježiti i reproducirati. Do fotografija individualca može se doći preko društvenih mreža ili ukradenih pametnih telefona te iz tih fotografija izgraditi modele lica ciljanih individualaca u svrhu prevare autentikacijskog sustava.

Sustavi za raspoznavanje lica su u širokoj upotrebi u svakodnevici kao način pojednostavljenja sigurne prijave u sustave (engl. *secure login*) ili kao sekundarne metode autentikacije u svrhu povećavanja sigurnosti [3].

Zbog toga ranjivost takvih sustava na napade lažiranjem (engl. *spoofing*) predstavlja veliki problem. Lažiranje lica (engl. *facial spoofing*) je metoda napada na sustave koji za autentikaciju koriste raspoznavanje lica. Ima za cilj prevariti takve sustave koristeći fotografije ili video snimke koje sadrže lica autoriziranih osoba te zloupotrijebiti time stečena prava autoriziranih korisnika u tom sustavu.

Napad lažiranjem lica može se izvesti na mnogo načina: koristeći isprintanu fotografiju lica, fotografiju prikazanu na ekranu, video koji sadrži lice, koristeći maske itd.

Većina dosadašnjih metoda za detekciju ovakvih napada temelji se na izrazima ili pokretima lica poput treptanja, pokreta glavom ili micanja usana. Međutim, sustav koji od korisnika zahtjeva ovakve radnje pri svakoj prijavi često je zamoran za korisnike. Zbog toga, razvoj tehnologija za detekciju napada lažiranjem lica se sve više usmjerava prema metodama koje ne zahtijevaju nikakve specifične akcije od korisnika.

3.1 Tipovi napada lažiranjem lica

Većina napada na sustave za autentikaciju koji se temelje na prepoznavanju lica spadaju u skupinu tzv. prezentacijskih napada (engl. *presentation attacks*) i dijele se na statičke i dinamičke. Statički i dinamički napadi dodatno se dijele na 2D i 3D napade.

	Statički	***Dinamički***
2D	fotografije, ravni papir ili maske, kao i transformacije tih objekata poput presavijanja, izrezivanja i sl.	reprodukcija videozapisa koji sadrži sliku lica na ekranu ili nekoliko fotografija u nizu
3D	3D ispisi, skulpture ili 3D maske	sofisticirani roboti za reprodukciju izraza lica, zajedno sa šminkom

Tablica 1: Kategorije prezentacijskih napada:

Tablica 1 navodi primjere metoda napada iz svake od kategorija prezentacijskih napada. Razvoj novih i boljih metoda za autentikaciju analizom lica prati i konstantan razvoj novih vrsta napada. Unatoč tome, 2D napadi su još uvijek dosta učestaliji od 3D napada jer su 3D napadi skuplji i zahtjevniji za izvršiti.

Sprječavanje lažiranje lica (engl. *face anti-spoofing*) je postupak provjere slike u svrhu utvrđivanja je li slika lica predstavljena autentikacijskom sustavu slika stvarne (žive) osobe ili je reproducirana ili sintetička, odnosno lažna [25].
Kada sustav za provjeru autentičnosti lica treba prepoznati lice osobe pomoću kamere i pripadajućeg softvera za raspoznavanje lica, važno je biti siguran da osoba koja traži provjeru autentičnosti zapravo predstavlja svoje lice kameri u vrijeme i mjesto zahtjeva za autentikaciju.

Suprotno tome, prevarant bi mogao pokušati predstaviti masku, fotografiju ili video zapis koji pokazuje kameri sliku legitimnog klijenta kako bi sustav napadača lažno ovjerio kao legitimnog. Ta vrsta prijetnje autentikacijskim sustavima općenito je poznata kao napad ponovnog ponavljanja (engl. *replay attack*).

3.2 Popularne metode za sprječavanje napada lažiranjem u sustavima za autentikaciju analizom izgleda lica

Budući da su 2D napadi najučestalija vrsta prezentacijskih napada, slijedi pregled popularnih rješenja za takve napade u sustavima za autentikaciju analizom izgleda lica. Pouzdano rješenje treba postići maksimalnu točnost, zahtijevati malo vremena i dati prednost korisničkom iskustvu.
Što je najvažnije, treba se integrirati s postojećim softverom za prepoznavanje lica.

3.2.1 Detekcija treptaja

Detekcija treptaja je test detekcije živosti koji je nevjerojatno precizan. Prirodno treptanje jednostavan je način utvrđivanja je li lice živo ili ne. Prosječni čovjek trepne 15–30 puta u minuti. Oči ostanu zatvorene oko 250 milisekundi tijekom treptaja. Moderne kamere snimaju videozapise s daleko manjim intervalima između kadrova od oko 50 milisekundi za kamere koje snimaju 30 okvira u sekundi (engl. *frames per second, fps*). Metoda je učinkovita za detekciju statičkih 2D prezentacjskih napda. U sklopu 2D dinamičkog prezentacijskog napda moguće je predočiti videozapis s licem osobe koja trepće autentikacijskom sustavu te ga na taj način prevariti.

3.2.2 Korištenje konvolucijskih neuronskih mreža

Duboko učenje i konvolucijske neuronske mreža (CNN) dodatna su rješenja koja mogu pomoći u borbi protiv prezentacijskih napada. Sprječavanje lažiranja lica može se shvatiti kao problem klasifikacije okvira videa u klase koje odgovaraju lažnim ili stvarnim slikama lica. Konvolucijska neuronska mreža može se naučiti da prepozna koje su stvarne fotografije, a koje lažne.

Temeljna pretpostavka jest da će mreža otkriti distorzije i izobličenja u lažnim slikama koja nisu vidljiva golom oku. Uspješnost ove metode uvelike ovisi o skupovima podataka na kojima je mreža učena i uvjetima u kojima su podatci prikupljani, a koji utječu na izgled slike uključujući značajke poput kvalitete kamere, okoliša, vibracija, osjvetljenja, itd. Međutim, zbog manjka skupova za učenje koji sadrže dovoljan broj podataka prikupljenih u različitim uvjetima (razine osvjetljenja, tip kamere i sl.) ova metoda se za sada pokazala održiva samo u slučajevima uske uporabe.

3.2.3 Tehnika izazov-odgovor (engl. *challenge-response technique*)

Izazovi i odgovori su još jedna učinkovita metoda za detekciju statičkih 2D prezentacijskih napada. Ova metoda koristi posebnu akciju sustava koja se zove *izazov* kojom se korisniku zadaje određeni zahtjev. Ukoliko korisnik u traženom vremenu ne izvrši zahtjev, odnosno ne da ispravan *odgovor* na traženi *izazov*, sustav će detektirati napad.

Izazovi mogu uključivati radnje kao što su:

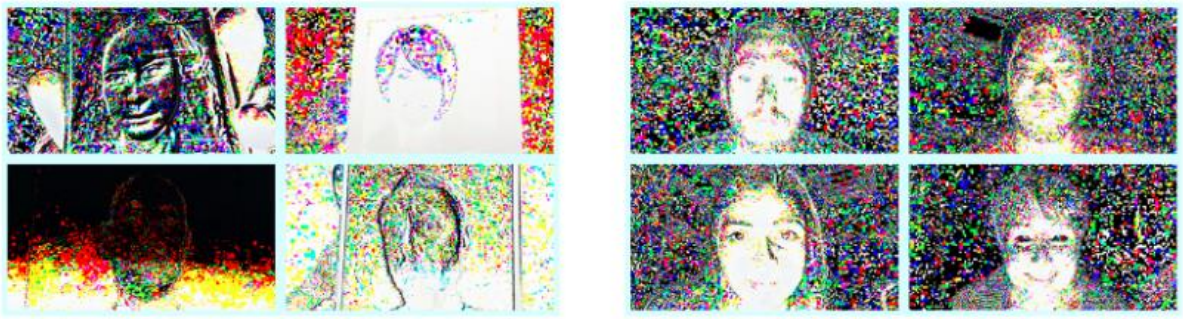
- osmijeh
- izrazi lica tuge ili sreće, iznenađenja ili ljutnje
- pokreti glave
- treptanje.

Međutim, iako je učinkovita - ova metoda zahtijeva sudjelovanje korisnika i može značajno utjecati na korisničko iskustvo.

3.2.4 Aktivni bljesak

Aktivni bljesak (engl. *active flash*) zanimljiva je i dosta obećavajuća tehnika.

Ovo rješenje omogućava otkrivanje prezentacijskih napada pomoću refleksije svjetla na licu. Ideja uključuje korištenje svjetlosnog okruženja koje se mijenja pomoću dodatnog svjetla koje dolazi sa zaslona uređaja.



Slika 3.1: Aktivni bljesak kao metoda detekcije prezentacijskih napada. Lijevo - lažne slike lica. Desno - stvarne slike lica[14]

Na ovaj način možemo razlikovati stvarna lica od lažnih.

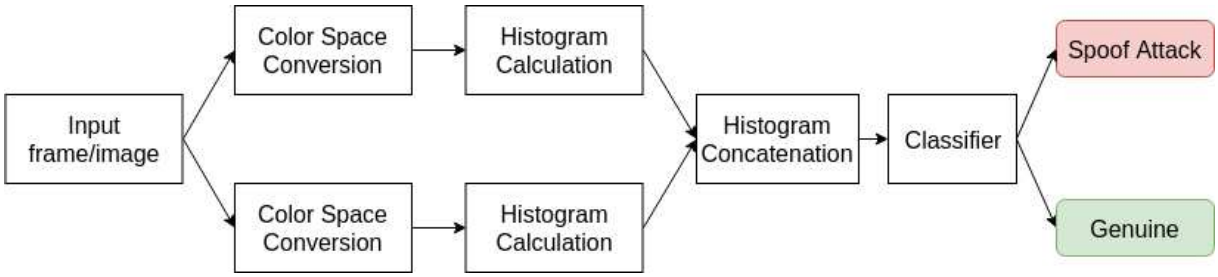
Uzimamo okvire prije i nakon što se dogodio bljesak i učimo neuronsku mrežu koristeći te podatke.

Moguće je izraditi model neovisan o kutu lica (s razumnim ograničenjima), pri čemu poravnavanje lica postaje nužno. Tehnologija bi se mogla učiniti sofisticiranijom na temelju posebnih slučajeva uporabe koje je potrebno riješiti. Problem ovakvih pristupa jest manjak dobrih skupova podataka na kojima bi se naučio model za detekciju lažiranja lica. Također, uspješnost metode može biti smanjena zbog vanjskih utjecaja, poput vibracija ili prevelike udaljenosti lica od zaslona, zbog čega refleksija svjetla ekrana na licu nije dovoljno snažna.

[14] Slika preuzeta s adrese: https://cutt.ly/8uiSHXf

3.2.5 Ansambl stabala odluke i CIE L*u*v i YCrCb prostori boja

Metoda za detekciju prezentacijskih napada predložena u [27] temelji se na konverziji slike iz RGB prostora boja u CIE L*u*v i YCrCb i računanju histograma slike lica u svakom od tih prostora. Nakon toga obavi se konkatenacija histograma i rezultantni vektor preda se na klasifikaciju ansamblu stabala odluke *ExtraTreesClassifier*, koji obavlja klasifikaciju vektora u 2 klase: lažne slike i stvarne slike lica.



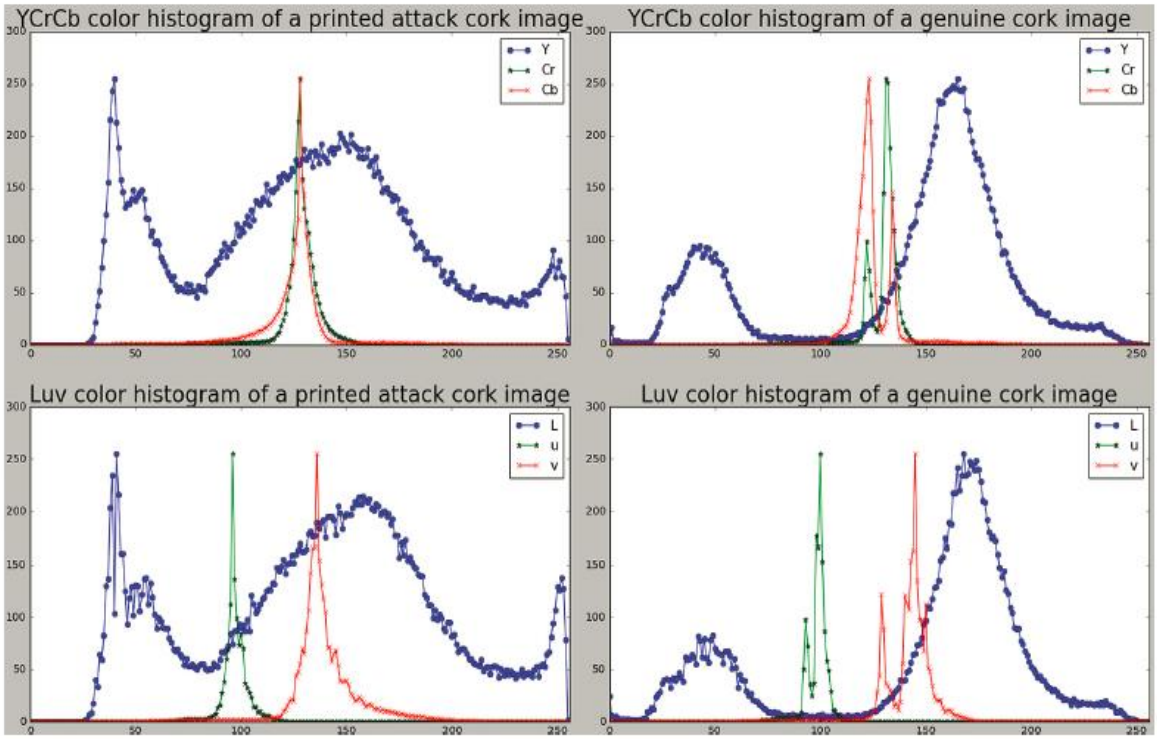
Slika 3.2: Metoda za detekciju prezentacijskih napada predložena u [27]

Unatoč učestalom korišenju RGB prostora boja u uređajima za snimanje i prikazivanje video zapisa, RGB nije najprikladniji prostor boja za otkrivanje napada lažiranja, zbog korelacije između crvene, zelene i plave boje koja ometa razdvajanje informacije o svjetlini i intezitetu boja.

Metoda predložena radu [27] koristi dva različita prostora boja: YCrCb i CIE L*u*v. Za svaku sliku u RGB prostoru boja, vrši se pretvorba u YCrCb i CIE L*u*v prostor boja. Zatim se izračunava 6 histograma, po jedan za svaku od komponenti ova dva prostora boja.
Dobivenih šest histograma spajaju se u vektor značajki (engl. *feature vector*) $\mathrm{FV} = (\mathrm{Y}, \mathrm{Cr}, \mathrm{Cb}, \mathrm{L}, \mathrm{u}, \mathrm{v})$ veličine 1536 (šest normaliziranih histograma u rasponu od 0 do 255) koji se predaje na ulaz klasifikatoru *ExtraTreesClassifier*. Konačno, klasifikator odlučuje odgovara li vektor značajki slici stvarnog objekta ili se radi o pokušaju napada.

Ideja za ovu metodu proizišla je iz promatranja razlika YCrCb i CIE L*u*v histograma slike stvarnog objekta i lažne slike objekta (objekt je bio pluteni čep). Autori rada [27]

primjetili su da informacija koja odgovara osvjetljenju na histogramima u oba prostora boja za fotografije stvarnog objekta ima gotovo "pravilan oblik" (što se ne događa na histogramima lažnih slika objekta). Što se tiče informacije o boji u oba spomenuta prostora boja na histogramima lažnih slike objekta pojavljuje se "pravilni oblik". Ovo ponašanje prikazano je na slici ispod.



*Slika 3.3: YCrCb i CIE L*u*v histogrami dvaju slika istog plutenog čepa. Lijevo su histogrami isprintane slike plutenog čepa, a desno histogrami stvarne (engl. genuine) slike čepa*[15]

Model predložen u [27] učen je na Print-attack [28] i Replay-attack [29] bazama podataka i imao je grešku EER 1.33% na evaluacijskom skupu Print-attack baze i 0.00756% na evaluacijskom skupu Replay-attack baze.

[15] Slika preuzeta iz [27]

3.2.6 3D kamere

3D kamere su najpouzdanije sredstvo protiv prezentacijskih napada. Precizne informacije o dubini piksela mogu pružiti visoku točnost prilikom prezentacijskih napada jer možemo razaznati razliku između lica i ravnog oblika.

3D napadi mogu uzrokovati poteškoće, no kamere su i dalje jedna od najpouzdanijih dostupnih tehnika protiv obrane lica. Međutim, usprkos dostupnosti kamera, nemaju ih svi korisnici na svojim računalima. Zbog toga je još uvijek u većini slučajeva praksa raditi s standardnim RGB slikama.

4. Eksperimenti

U okviru rada izrađen je sustav za autentikaciju osoba analizom izgleda lica koji radi u stvarnom vremenu. Proces autentikacije može se podijeliti na 3 osnovna koraka:

1. detekciju lica u svakom okviru videa
2. detekciju prezentacijskih napada
3. verifikaciju lica.

Za testiranje sustava izgrađena je desktop aplikacija za autentikaciju osobe koja u stvarnom vremenu obavlja gore navedene korake. Tekst na korisničkom sučelju aplikacije pisan je na engleskom jeziku.

Nakon pokretanja procesa autentikacije pali se web-kamera računala i započinje detekcija lica koja se obavlja na svakom okviru videa kojeg kamera snima.

Zatim se pokreće detekcija napada. Detekcija napada obavlja se u dvije faze: najprije se od korisnika traži da izvrši određenu radnju (npr. treptanje). Ukoliko se ne detektira ispravan odgovor na traženi zahtjev (korisnik nije izvršio traženu radnju), detektira se napad. Ako se detektira ispravan odgovor (korisnik je izvršio traženu radnju), pokreće se druga faza detekcije napada koja na temelju karakteristika histograma slika snimanih web-kamerom detektira pokušaj napada.

Ako se ni tada ne detektira napad, sa slike dobivene kamerom izreže se dio na kojem je detektirano lice i takva slika predaje se modelu za verifikaciju lica. Model na temelju slike računa vektor značajki lica. Zatim se taj vektor značajki uspoređuje sa svim vektorima značajki lica autoriziranih korisnika na način da se računa euklidska udaljenost između vektora. Ako je najmanja izračunata udaljenost manja od unaprijed definiranog praga, verifikacija lica je uspješno obavljena i na ekranu se ispisuje poruka dobrodošlice. Ako najmanja izračunata udaljenost nije manja od unaprijed definiranog praga, sustav zaključuje da osoba ispred kamere nije autorizirani korisnik te ispisuje prikladnu poruku na ekranu aplikacije.

U sljedećim poglavljima opisane su implementacije i rezultati svih komponenti sustava, kao i primjeri rada testne aplikacije.

4.1 Programska podrška

4.1.1 Python

Python je programski jezik otvorenog koda, opće namjene, interpretiran i visoke razine, koji je nastao 1990. godine.

Podržava više programskih paradigmi – objektno orijentirano, strukturno i aspektno orijentirano programiranje.

Python je jedan od najpopularnijih programskih jezika korištenih u području strojnog i dubokog učenja. Jednostavna i izražajna sintaksa, kao i čitljivost samog jezika, omogućavaju brzu implementaciju i testiranje složenih procesa.

Nadalje, Python nudi mnoštvo besplatnih biblioteka za strojno i duboko učenje koje pojednostavljuju razvojne troškove i skraćuju vrijeme razvoja. Neke od najopularnijih uključuju NumPy, Scipy, Matplotlib, TensorFlow, Scikit-learn, Keras, PyTorch i mnoge druge.

U sklopu ovog rada korištena je verzija Pythona 3.6.9.

4.1.2 Google Colaboratory

Google Colaboratory[16], ili popularnije Colab, besplatni je servis koji omogućava jednostavno pisanje i izvođenje Python programa u sklopu Jupyter bilježnica. Kod se izvodi na virtualnim strojevima u oblaku (engl. *cloud*), koji dolaze s unaprijed instaliranim Pythonom i svim popularnim bibliotekama, tako da za korištenje Colaba najčešće nije potrebno nikakvo ili samo minimalno dodatno podešavanje okoline.

Nadalje, Colab nudi mogućnost korištenja GPU procesora. GPU procesori (engl. *graphics processing unit*) nude znatno hardversko i softversko ubrzanje u odnosu na CPU procesore, pogodno za probleme dubokog učenja čija računska složenost proizlazi iz matričnih operacija. Korištenje Colab GPU-a je besplatno uz ograničenje kontinuiranog korištenja od maksimalno 12 sati.

[16] Službena adresa za Google Colaboratory: https://colab.research.google.com/notebooks/intro.ipynb

U sklopu Colaba nudi se nekoliko modela GPU-a (Nvidia K80s, T4s, P4s and P100s), čija dostupnost varira s obzirom na vrijeme korištenja procesora i ukupan broj korisnika Colaba koji u određenom trenutku imaju potražnju za GPU-om. Model GPU-a se automatski dodjeljuje korisniku pri spajanju na Colab okolinu i ne može se ručno odabrati.

Google Colaboratory korišten je u sklopu ovog rada kao okolina za učenje svih konvolucijskih neuronskih mreža iz eksperimentalnog dijela zadataka.

4.1.3 Dlib

Dlib[17] je besplatna biblioteka otvorenog koda, pisana u C++ programskom jeziku, koja sadrži implementacije različitih algoritama strojnog učenja za probleme klasifikacije, regresije, grupiranja, transformacije podataka i mnoge druge. Dlib sadrži mnoge gotove, već naučene modele, mnoštvo primjera za korištenje alata koje nudi te detaljnu dokumentaciju svih klasa i funkcija koje sadrži.
Biblioteka je višeplatformska (engl. *cross-platform*), može se koristiti u raznim aplikacijama, na raznim operacijskim sustavima.
U sklopu ovog rada korišten je gotovi model iz dlib biblioteke za detekciju 68 karakterističnih točaka lica.

4.1.4 Opencv

OpenCV[18] (engl. *Open Source Computer Vision*) je skup biblioteka otvorenog koda. Biblioteke su višeplatformske, mogu se koristiti u raznim aplikacijama, na raznim operacijskim sustavima. Osnovna zadaća OpenCV-a je jednostavno izvođenje zadataka iz područja računalnog vida i omogućavanje jednostavnog i brzog razvoja algoritama i aplikacija koje koriste računalni vid. Zbog kompliciranih algoritama koji

[17] Službena adresa za dlib: http://dlib.net/

[18] Službena adresa za OpenCV: https://opencv.org/

zahtjevaju veliki broj složenoh operacija i rade s velikim količinama podataka, zahtjeva se optimiziran kod napisan na nižoj razini radi veće brzine izvršavanja. Zbog toga su algoritmi OpenCV biblioteka originalno pisani u C programskom jeziku.
U sklopu ovog rada korištena je verzija 4.2.0 OpenCV biblioteke.

4.1.5 TensorFlow

TensorFlow[19] je biblioteka za za numeričko računanje i primjenu strojnog učenja.
Biblioteka je postala javno dostupna za korištenje 9. studenog 2015.
TensorFlow se može koristiti za izradu i učenje dubokih neuronskih mreža za klasifikaciju slika, rukom pisanih znamenki, obradu prirodnog jezika i za moge druge primjene. Osigurava izvrsnu podršku u arhitekturi, kako bi se omogućilo jednostavno korištenje na širokom rasponu platformi, uključujući osobna računala, servere i mobilne uređaje.
Apstrakcija je glavna prednost TensorFlow Python-a. Ova značajka omogućuje programerima da se usredotoče na sveobuhvatnu logiku aplikacije, umjesto da se bave implementacijskim detaljima algoritama. Pomoću ove biblioteke mogu se jednostavno implementirati modeli za duboko i strojno učenje i stvoriti jedinstvene responzivne aplikacije, koje reagiraju na korisničke unose poput izraza lica ili glasa.
U sklopu ovog rada korištena je verzija 2.1.0 TensorFlow biblioteke.

4.1.6 Keras

Keras[20] je Python biblioteka otvorenog koda napisana za izgradnju neuronskih mreža i projekata strojnog učenja. Može se izvoditi na Deeplearning4j, MXNet, Microsoft Cognitive Toolkit (CNTK), Theano ili TensorFlow backendu. Nudi gotovo sve samostalne module uključujući optimizatore, neuronske slojeve, funkcije aktiviranja,

[19] Službena adresa za TensorFlow: https://www.tensorflow.org/

[20] Službena adresa za Keras: https://keras.io/

sheme inicijalizacije, funkcije gubitka i sheme reguliranja. Kako je model već definiran u kodu, zasebne konfiguracijske datoteke modela nisu potrebne.
Keras olakšava dizajn i razvoj neuronske mreže. Sadrži algoritme za normalizaciju, optimizacijski sloj i slojeve za aktiviranje. Umjesto da bude krajnja biblioteka za strojno učenje Python-a, Keras funkcionira kao korisničko, proširivo sučelje koje povećava modularnost i potpunu izražajnost.
U sklopu ovog rada korištena je verzija 2.3.1 Keras biblioteke.

4.1.7 Scikit-learn

Scikit-learn[21] je još jedna istaknuta Python biblioteka za strojno učenje s širokim rasponom algoritama za grupiranje, regresiju i klasifikaciju. Jednostavno se koristi s brojčanim i znanstvenim knjižnicama Pythona poput NumPy i SciPy.
Ova Python biblioteka može se koristiti za zadatake nadziranog i nenadziranog strojnog učenja.
U sklopu ovog rada korišten je klasifikator ExtraTreesClassifier, ansambl stabala odluke iz scikit-learn biblioteke, za detekciju dinamičkih prezentacijskih napada.

4.1.8 Kivy

Kivy[22] je besplatna Python biblioteke za razvoj mobilnih aplikacija i drugog aplikacijskog softvera s prirodnim korisničkim sučeljem. Distribuira se pod uvjetima MIT licence, a može se izvoditi na Androidu, iOS-u, GNU / Linuxu, OS X i Windows-u.
U klopu ovog rada korištena je za izradu grafičkog sučelja aplikacije za testiranje implementacije.

[21] Službena adresa za Scikit-learn: https://scikit-learn.org/stable/

[22] Službena adresa za Kivy : https://kivy.org/doc/stable/api-kivy.html

4.2 Skupovi podataka

4.2.1 MS1M-ArcFace

MS-Celeb-1M skup sadrži 10 milijuna slika lica približno 100 000 osoba, prikupljenih na internetu u svrhu učenja modela za raspoznavanje lica.

MS1M-ArcFace[23] skup podataka podskup je Microsoft Celeb (MS-Celeb-1M) skupa podataka, izrađen u sklopu rada [20], gdje je korišten kao skup za učenje modela. Iz MS-Celeb-1M skupa izbačene su sve slike sadržane u skupovima slika korištenih za evaluaciju naučenih modela u [20], a na zadržanim slikama korišteno je poravnavanje metodom MTCNN[24].

MS1M-ArcFace sadrži 85 742 klase i ukupno 5 822 653 slike dimenzija (112, 112, 3).

4.2.2 LFW

LFW[25] (engl. *Labeled Faces in the Wild*) skup podataka sadrži 5749 klasa (ukupno 13233 slike), od kojih je 1680 zastupljeno sa dvije ili više slika u ukupnom skupu.

LWFskup koristi se kao benchmark skup za verifikaciju modela za raspoznavanje lica s otvorenim skupom (Slika 2.9).

Slijedeći metodu opisanu u [20], u ovom radu korištena je inačica[26] LFW skupa koja sadrži slike dimenzija (112, 112, 3) poravnate metodom MTCNN.

Slika 4.1: Primjeri slika iz LFW skupa podataka

[23] MS1M-ArcFace skup može se preuzeti iz Dataset-Zoo u sklopu službenog Github repozitorija rada [20] na adresi: https://github.com/deepinsight/insightface/wiki/Dataset-Zoo

[24] MTCNN alignement: https://kpzhang93.github.io/MTCNN_face_detection_alignment/index.html

[25] Službena adresa za LFW: http://vis-www.cs.umass.edu/lfw/

[26] Skup LFW-align-112 može se preuzeti s adrese: https://drive.google.com/file/d/1WO5Meh_yAau00Gm2Rz2Pc0SRldLQYigT/view

4.2.3 FER2013

FER2013[27] je skup podataka za učenje modela za raspoznavanje izraza lica (engl. *face expression recognition*). Sastoji se od 35 887 sivih slika dimenzija (48,48) svrstanih u 7 klasa. Slika 4.2 prikazuje primjere slika svake od klasa.

Slika 4.2: Primjeri slika svake od klasa iz skupa FER2013

Tablica 2 prikazuje oznake i nazive klasa te zastupljenost svake klase unutar skupa.

Oznaka klase	Emocija (izraz lica)	Broj slika klase unutar skupa
0	Ljutnja (engl. *anger*)	4593
1	Gađenje (engl. *disgust*)	547
2	Strah (engl. *fear*)	5121
3	Sreća (engl. *happy, happiness*)	8989
4	Tuga (engl. *sad, sadness*)	6077
5	Iznenađenje (engl. *surprise*)	4002
6	Neutralno (engl. *neutral*)	6198

Tablica 2: Zastupljenost pojedinih klasa u FER2013 skupu podataka

[27] Skup FER2013 preuzet s adrese: https://www.kaggle.com/deadskull7/fer2013

4.2.4 CK+48

CK+48 (Extended Cohn-Kanade Dataset) je skup podataka za učenje modela za raspoznavanje izraza lica (engl. *face expression recognition*). Skup korišten u sklopu ovog rada preuzet je s Kaggle stranice[28] i sadrži 981 sivu sliku dimenzija (48,48) svrstanih u 7 klasa. Slika 4.3 prikazuje primjere slika svake od klasa.

Slika 4.3: Primjeri slika svake od klasa iz skupa CK+48

Tablica 2 prikazuje nazive klasa i korištene oznake u sklopu ovog rada te zastupljenost svake klase unutar skupa.

Oznaka klase	Emocija (izraz lica)	Broj slika klase unutar skupa
0	Ljutnja (engl. *anger*)	135
1	Gađenje (engl. *disgust*)	177
2	Strah (engl. *fear*)	75
3	Sreća (engl. *happy, happiness*)	207
4	Tuga (engl. *sad, sadness*)	84
5	Iznenađenje (engl. *surprise*)	249
6	Prijezir (engl. *contempt*)	54

Tablica 3: Zastupljenost pojedinih klasa u CK+48 skupu podataka

[28] CK+48 skup podataka preuzet s adrese: https://www.kaggle.com/shawon10/ckplus

4.2.5 LCC-FASD

LCC-FASD[29] je skup podataka namijenjen za učenje modela za problem detekcije prezentacijskih napada, prvenstveno dinamičkih 2D napada. Skup sadrži ukupno 18827 slika lica, od čega je 1942 slika stvarnog lica, a 16885 slika su lažne slike lica. Lažne slike dobivene su kao snimke stvarnih lica kamerama visoke rezolucije. Skup podataka prikupljen je uz pomoć servisa Youtube, Amazon Mechanical Turk i Yandex Toloka i predstavljen u sklopu rada [3].

Dimenzije slika nisu ujednačene u cijelom skupu – autori [3] nisu mijenjali veličinu slike (engl. *resizing*) da se prilikom interpolacije ne bi izgubile potencijalno bitne karakteristike slike. Apsolutna veličina slika varira između 150 i 1350 piksela.

Bitna značajka skupa koju ističu autori [3] jest velik raspon modernih uređaja korištenih za dobivanje snimki lažnih lica. Dobivene su tako da su se različitim uređajima snimale slike stvarnog (engl. *genuine*) lica prikazane na ekranu nekog drugog uređaja (korišteno je ukupno 82 različita uređaja za **prikaz** stvarnih lica).
Nadalje, korištena su 83 različita uređaja (većinom različiti modeli pametnih telefona) za **snimanje** slika lažnih lica.
Skup je podijeljen u 3 podskupa: podskup za učenje, testni podskup i podskup za evaluaciju modela, kako prikazuje Tablica 4.

Podskup	Broj identiteta	Broj stvarnih lica	Broj lažnih lica
Učenje	118	1223	7076
Test	25	405	2543
Evaluacija	100	214	7266
UKUPNO	243	1942	16885

Tablica 4: LCC-FASD podjela na podskupove za učenje, test i evaluaciju

[29] LCC-FASD skup podataka preuzet sa službene adrese:
https://drive.google.com/file/d/1NeyTFAwdJSjxA9ZtdviwdUjdptEVjM_i/view

Slika 4.4: LCC-FASD - primjeri slika stvarnog lica (4 lijeve slike) i lažnog lica (4 desne slike) iz [3]

4.3 Implementirane metode za detekciju napada lažiranjem lica

U sklopu ovog rada implementirane su 3 metode za detekciju napada lažiranja lica:

1. detekcija treptaja
2. tehnika izazov-odgovor
3. ansambl stabala odluke i CIE L*u*v i YCrCb prostor boja.

Prve dvije metode namijenjene su detekciji statičkih 2D prezentacijskih napada. Tehnika izazov-odgovor sastoji se od detekcije treptaja i raspoznavanja izraza lica, dakle prva metoda implementirana je kao komponenta druge metode. Princip rada je sljedeći: sustav korisniku zada izazov (*„trepnite”, „nasmiješite se”*). Ukoliko korisnik ne obavi zatraženu radnju u vremenskom roku od 2 sekunde nakon što ju je sustav zadao, sustav to prepoznaje kao pokušaj napada i prekida postupak autentikacije.

Treća metoda namjenjena je detekciji dinamičkih 2D napada. Ako je napadač upoznat s činjenicom da sustav koristi prve dvije metode za detekciju napada lažiranjem, može izvršiti napad tako da na svom uređaju ima softver koji je sposoban detektirati zatraženi izazov i generirati ispravan odgovor (video sekvencu lica u kojoj se obavlja tražena

radnja), odnosno izvršiti dinamički 2D prezentacijski napad. Zbog toga je implementirana i dodatna metoda detekcije napada lažiranja pomoću ansambla stabala odluke.

4.3.1 Detekcija treptaja

Detekcija treptaja obavlja se u 4 koraka:

1. detekcija lica
2. računanje karakterističnih točaka lica
3. računanje EAR (engl. *eye aspect ratio*) vrijednosti [30]
4. donošenje odluke jesu li oči zatvorene ili otvorene na temelju izračunate EAR vrijednosti i unaprijed definiranog praga za EAR.

Implemetacija detekcije treptanja temelji se na internetskom članku [36] objavljenom na blogu PyImageSearch.

Za zadatak detekcije lica razmatrana su dva detektora:

1. detektor iz dlib biblioteke temeljen na HOG (engl. *Histogram Oriented Gradients*) i SVM (engl. *Support Vector Machine*) algoritmima i koji se inicijalizira na sljedeći način:

```
detector = dlib.get_frontal_face_detector()
```

2. detektor iz OpenCv biblioteke namjenjen za detekciju lica okrenutih direktno prema kameri, koji se inicijalizira na sljedeći način:

```
faceCascade =
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")[30]
```

Nakon testiranja obaju detektora u sklopu testne aplikacije izrađene za testiranje rada

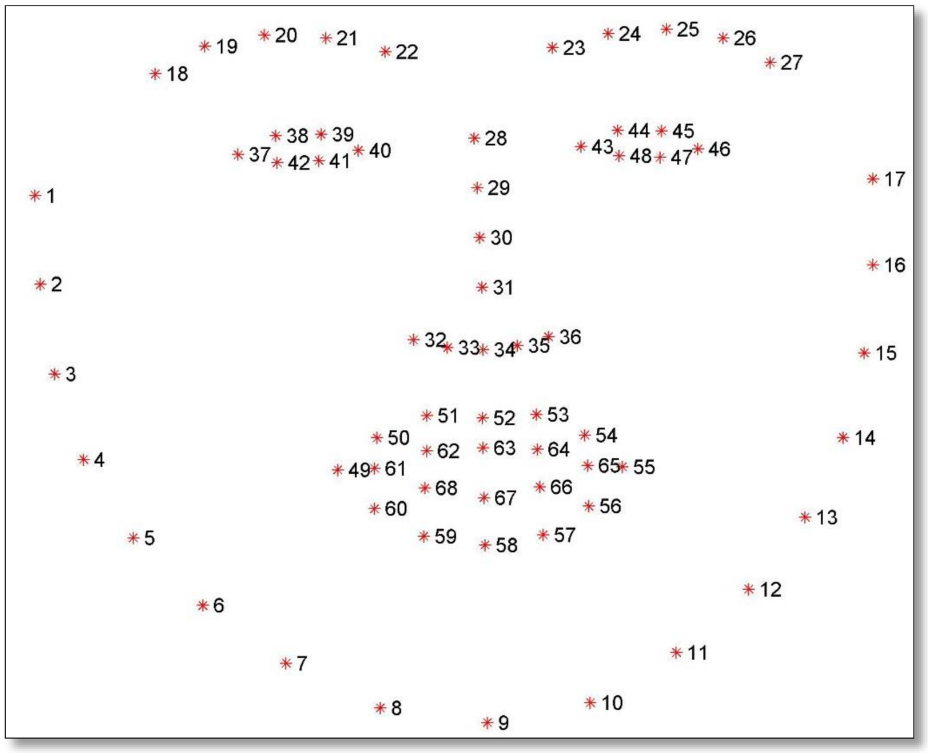
[30] Datoteka haarcascade_frontalface_default.xml može se preuzeti s adrese: https://github.com/opencv/opencv/tree/master/data/haarcascades

cijelog sustava za autentikaciju, pokazalo se da je detektor iz OpenCV biblioteke znatno brži i pogodniji za korištenje u sklopu spomenute aplikacije koja radi u stvarnom vremenu.

Za zadatak pronalaženja 68 karakterističnih točaka lica (engl. *face landmarks*) korišten je gotovi (naučeni) model iz dlib biblioteke koji se inicijalizira na sljedeći način:

```
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```[31]

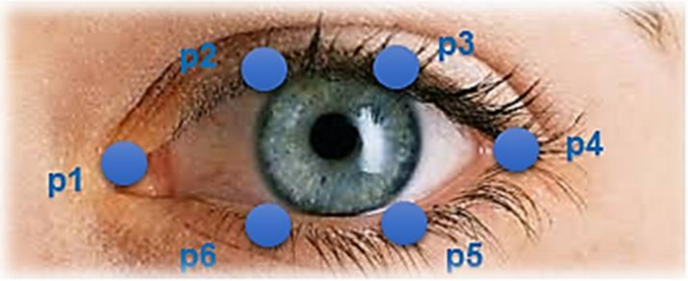
Na slici ispod prikazane su karakteristične točke lica čije koordinate pronalazi ovaj model.



Slika 4.5: Dlib - 68 karakterističnih točaka lica

[31] Datoteka shape_predictor_68_face_landmarks.dat može se preuzeti s adrese: http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

Nakon pronalaska karakterističnih točaka lica, izdvajaju se koordinate očiju potrebne za detekciju treptanja. Svako oko definirano je sa 6 karakterističnih točaka, odnosno 6 (x,y) koordinata, kako je prikazano na slici ispod.



Slika 4.6: Karakteristične toče oka

Vrijednost EAR definirana je kao:

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2 * \|p_4 - p_1\|}.$$

U brojniku se računaju euklidske udaljenosti između vertikalnih karakterističnih točaka oka, a u nazivniku između horizontalnih. Kada je oko otvoreno, ovaj omjer udaljenosti među karakterističnim točkama oka je približno jednak. Pri treptaju brzo pada prema nuli.

Na tom opažanju temelji se metoda detekcije treptaja u stvarnom vremenu. EAR vrijednost računa se za svaki okvir videa u kojem je prisutno lice na način da se izračuna EAR vrijednost za lijevo i desno oko, i zatim izračuna njihova srednja vrijednost. Dakle, $frame_{ear} = \frac{left\_eye_{ear} + right\_eye_{ear}}{2}$.

Ako je izračunata vrijednost manja od unaprijed definiranog praga, sustav pamti taj događaj kao treptaj. U ovom radu definiran je prag $ear\_threshold = 0.2$.

Nadalje, definiran je još jedan uvjet koji utječe na konačnu odluku sustava o tome je li se dogodio treptaj ili ne: broj uzastopnih okvira videa u kojima je $\mathrm{frame_{ear}} < ear\_threshold$ mora biti veći ili jednak unaprijed definiranoj vrijednosti. U ovom radu korištena je konstanta $ear\_consecutive\_frames = 3$. Dakle, ukoliko je zadovoljen uvjet $\mathrm{frame_{ear}} < ear\_threshold$ u 3 uzastopna okvira videa, detektiran je treptaj.

Evaluacija algoritma i pronalazak odgovarajućih vrijednosti za gore spomenute konstante obavljeni su isprobavanjem različitih vrijednosti za pragove uz pokretanje algoritma nad sekvencom okvira snimljenih web-kamerom laptopa u stvarnom vremenu.

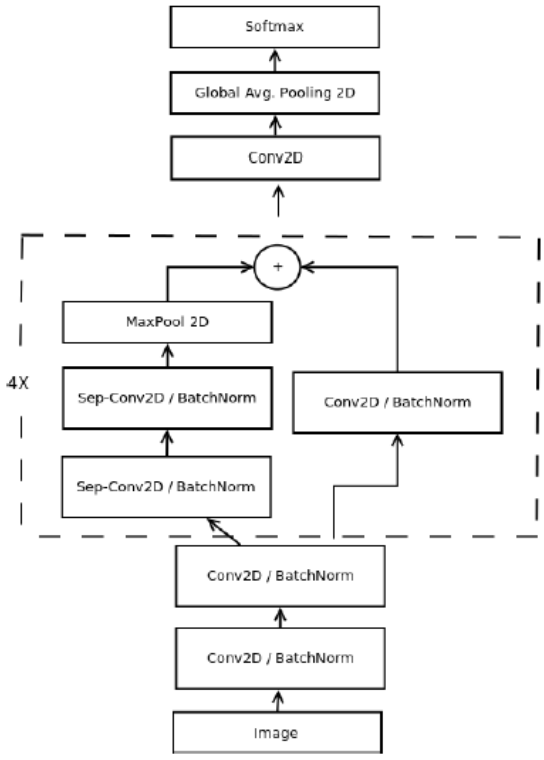
Broj uzastopnih okvira u kojima je zadovoljen uvjet $\mathrm{frame}_{\mathrm{ear}} < ear\_threshold$ odgovara onom definiranom u [36] i iznosi 3.

Vrijednost praga $ear\_threshold = 0.2$ pokazala se bolja u praksi nego ona predložena u [36]. Prevelika vrijednost praga može dovesti do lažnih detekcija u situacijama kada su oči poluzatvorene ili pogled nije uperen direktno prema kameri (npr. glava ili samo pogled su usmjereni lagano prema gore ili dolje).

4.3.2 Raspoznavanje izraza lica

Raspoznavanje izraza lica implementirano je slijedeći metodu predloženu u [31].

Konvolucijska neuronska mreža korištena za zadatak raspoznavanja izraza lica je Mini_Xception, čija je arhitektura definirana u [31] i prikazana na slici ispod.



Slika 4.7: Mini_Xception arhitektura predložena u radu [31]

Mini_Xception mreža naučena je na FER2013 skupu podataka, s tim da su dimenzije slika promijenjene sa (48, 48) na (64, 64). Nakon pretprocesiranja podataka, skup podataka podijeljen je u podskup za učenje i testni podskup, gdje u testni skup spada 20% skupa Fer2013. Na slici ispod prikazan je isječak koda koji sadrži metodu $preprocess\_input(x)$ korištenu za pretprocesiranje podataka.

```
46 def preprocess_input(x):
47     x = x.astype('float32')
48     x = x / 255.0
49     x = x - 0.5
50     x = x * 2.0
51     return x
```

Slika 4.8: Metoda za pretprocesiranje podataka iz FER2013 skupa

Slika 4.9 prikazuje kod korišten za učenje modela. Temelji se na kodu s Github repozitorija *face_classification* [38] autora rada [31]. Linije 2-7 definiraju instancu ImageDataGenerator klase iz Keras biblioteke koja se koristi pri učenju modela za augmentaciju podataka. Pri učenju, nakon uzorkovanja jedne mini-grupe podatka (engl. *mini-batch*) podatci unutar te grupe se nasumično mijenjaju, čime se osigurava da mreža u svakoj epohi učenja dobije drugačije podatke. U ovoj implementaciji promjene koje se vrše nad podatcima su rotacija, horizontalni i vertikalni pomaci, zumiranje i horizontalno zrcaljenje slike.

```
[ ]  1 # data generator
     2 data_generator = ImageDataGenerator(
     3                         rotation_range=10,
     4                         width_shift_range=0.1,
     5                         height_shift_range=0.1,
     6                         zoom_range=.1,
     7                         horizontal_flip=True)
     8
     9 # model parameters/compilation
    10 model = mini_XCEPTION(input_shape, num_classes)
    11 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])
    12
    13
    14 # callbacks
    15 log_file_path = base_path + 'logs/' + 'xception_emotion_training.log'
    16 csv_logger = CSVLogger(log_file_path, append=False)
    17 early_stop = EarlyStopping('val_loss', patience=20)
    18 reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1, patience=12, verbose=1, min_lr=0.00001)
```

Slika 4.9: Kod – učenje modela Mini_Xception

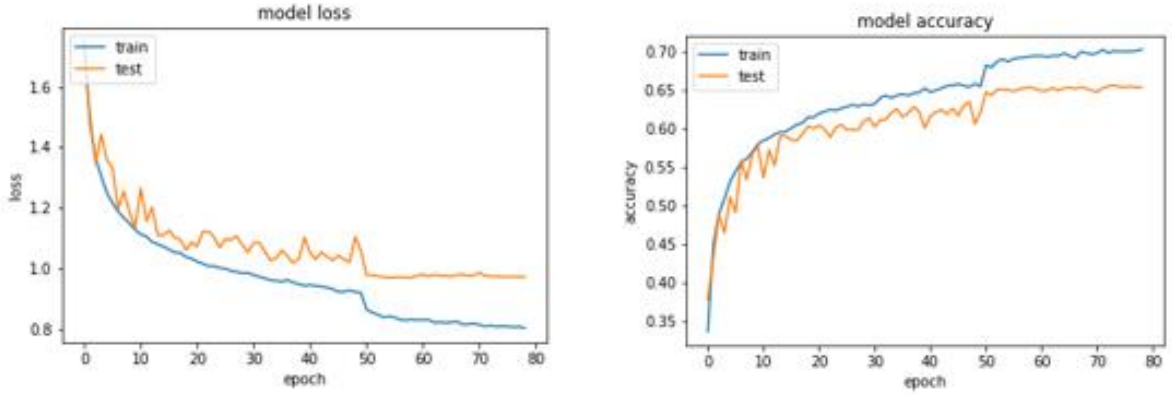
Tablica 5 prikazuje hiperparametre modela. Tokom učenja korišten je *ReduceLROnPlateau* callback iz biblioteke Keras koji smanjuje stopu učenja za faktor 0.1 ako se gubitak na testnom skupu ne smanji kroz 12 uzastopnih epoha. Nadalje, korišten je i callback *EarlyStopping* iz biblioteke Keras koji zaustavlja učenje modela ako se gubitak na testnom skupu ne smanji kroz 20 uzastopnih epoha.

Hiperparametar	Vrijednost
Optimizator	**Adam (Keras)**
Početna stopa učenja	**0.001**
Maksimalan broj epoha	**1000**
Gubitak	**Unakrsna entropija**
Veličina mini-grupe (engl. *batch size*)	**32**

Tablica 5: Hiperparametri modela Mini_Xception

Usporedba s rezultatima iz literature

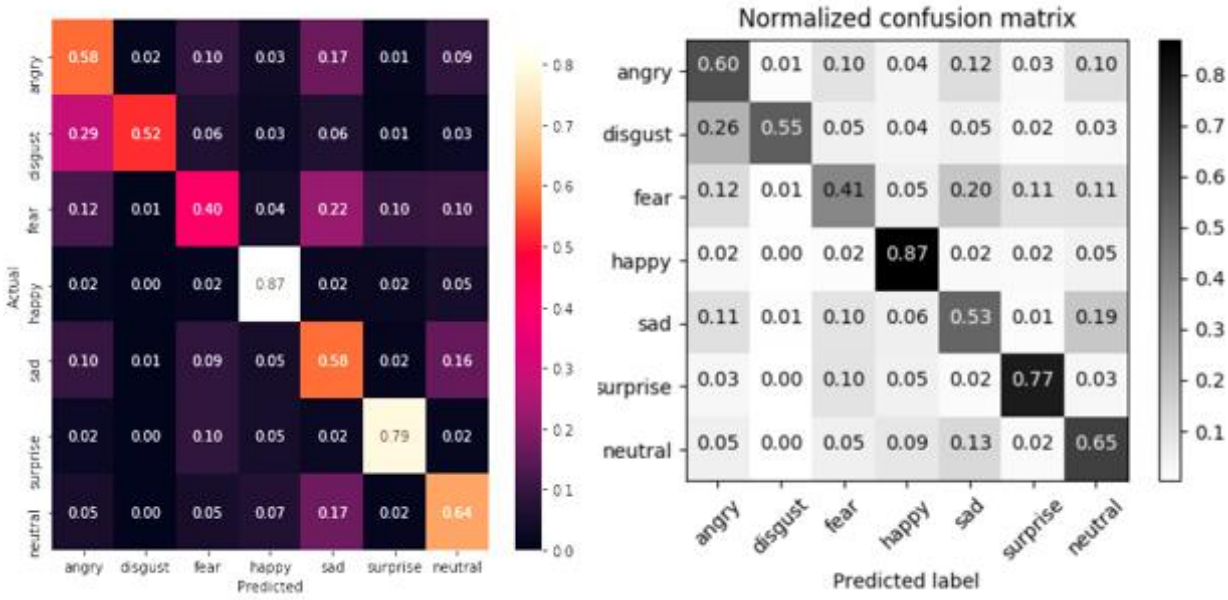
Učenje Mini_Xception mreže trajalo je 80 epoha. Postignuta točnost (engl. *accuracy*) na testnom skupu je 65.41%. Rad [31] navodi postignutu točnost od 66% za model naučen za raspoznavanje emocija na FER-2013 skupu podataka.



Slika 4.10: Gubitak i točnost na skupu za učenje i testnom skupu - Mini_Xception, FER2013

Na slici ispod prikazana je usporedba matrice konfuzije modela naučenog u sklopu ovog rada i matrice konfuzije dane u radu [31] na FER2013 skupu podataka.

Matrica konfuzije prikazuje performanse modela na testnom podskupu FER2013 koji je korišten za validaciju modela pri učenju.



Slika 4.11: Usporedba matrica konfuzije modela naučenog u sklopu ovog rada i modela iz [31]

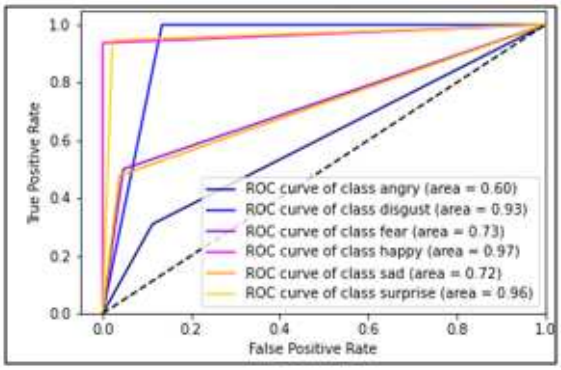
Iz matrica konfuzije vidljivo je da su performanse modela naučenog u sklopu ovog rada gotovo jednako dobre kao i one navedene u [31], s tim da je broj točno klasificiranih primjera klase „happy“ jednak kao i u [31], a broj klasificiranih primjera klasa „sad“ i „surprise“ veći nego u [31]. Međutim, razlike su jako male i vjerojatno proizlaze iz različitog načina uzorkovanja testnog skupa iz cijelog FER2013 skupa podataka.

Evaluacija modela

Evaluacija modela obavljena je na CK+48 skupu podataka, s tim da su dimenzije slika promijenjene sa (48, 48) na (64, 64). Pretprocesiranje je obavljeno na isti način kao i na skupu FER2013. Skupovi FER2013 i CK+48 oba imaju 7 klasa, ali je od tih 7 samo 6 jednako: ljutnja, gađenje, strah, sreća, tuga i iznenađenje. Sedma klasa u FER2013 skupu je neutralan izraz lica, a u CK+48 skupu jest prijezir. Zbog tog su za evaluaciju korišteni samo podatci koji pripadaju klasama koje sadrže oba skupa.

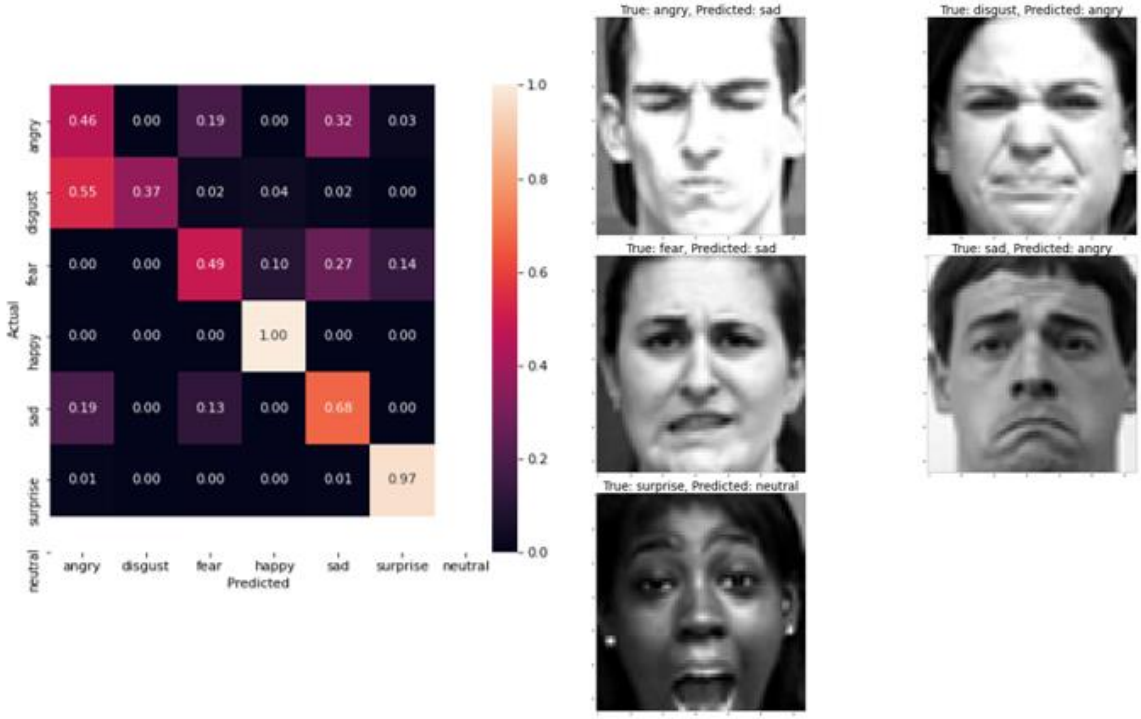
Rezultati klasifikacije dobiveni su korištenjem *classification_report* metrike iz biblioteke Scikit-learn i prikazani na slici ispod.

	precision	recall	f1-score	support
angry	0.31	0.36	0.33	135
disgust	1.00	0.34	0.51	177
fear	0.50	0.47	0.48	75
happy	0.94	1.00	0.97	207
sad	0.46	0.63	0.54	84
surprise	0.95	0.94	0.94	249
neutral	0.00	0.00	0.00	0
accuracy			0.69	927
macro avg	0.60	0.53	0.54	927
weighted avg	0.78	0.69	0.70	927



Slika 4.12: Mini_Xception - classification_report na skupu CK+48 bez podataka klase 'prijezir' (lijevo); ROC-AUC krivulje - Mini_Xception. CK+48 (desno)

Slika 4.13 prikazuje ROC-AUC krivulje i matricu konfuzije dobivene evaluacijom modela na skupu CK+48 dobivene korištenjem metrika *roc_curve, auc i confusion_matrix* iz biblioteke Scikit-learn i neke pogrešno klasificirane primjere.



Slika 4.13: Matrica konfuzije modela Mini_Xception na skupu CK+48 (lijevo); Primjer pogrešno klasificiranih slika (desno)

4.3.3 Detekcija dinamičkih 2D prezentacijskih napada

Za detekciju dinamičkih 2D prezentacijskih napada implementirana je metoda opisana u poglavlju 3.2.5 koja se temelji na konverziji slike iz RGB prostora boja u CIE L*u*v i YCrCb i računanju histograma slike lica u svakom od tih prostora. Nakon toga obavi se konkatenacija histograma i rezultantni vektor preda se na klasifikaciju ansamblu stabala odluke *ExtraTreesClassifier.*

Detekcija napada zamišljena je kao klasifikacija slike lica u jednu od dvije klase:

- klasa 0 – stvarno lice (engl. *genuine face*)
- klasa 1 – lažno lice (engl. *spoofed face*).

Slijedeći metodu opisanu u [27] korišten je klasifikator *ExtraTreesClassifier* iz Scikit-learn biblioteke. Klasifikator je učen na LCC-FASD skupu podataka.

LCC-FASD podijeljen je u 3 podskupa: podskup za učenje, testni podskup i podskup za evaluaciju modela.

Slika 4.14: Struktura direktorija skupa LCC-FASD (lijevo) i npy datoteka sa vektorima značajki(desno)

Kako je vidljivo sa Slika 4.14, slike skupa za učenje organizirane su u poddirektorije s obzirom na klasu kojoj pripadaju. Nazivi slika imaju sljedeći format: *<model_uređaja>_<identifikator_osobe>_<scena>_<broj_okvira_videa>.png*, kao na primjer *SMG950U_id123_s0_120.png*.

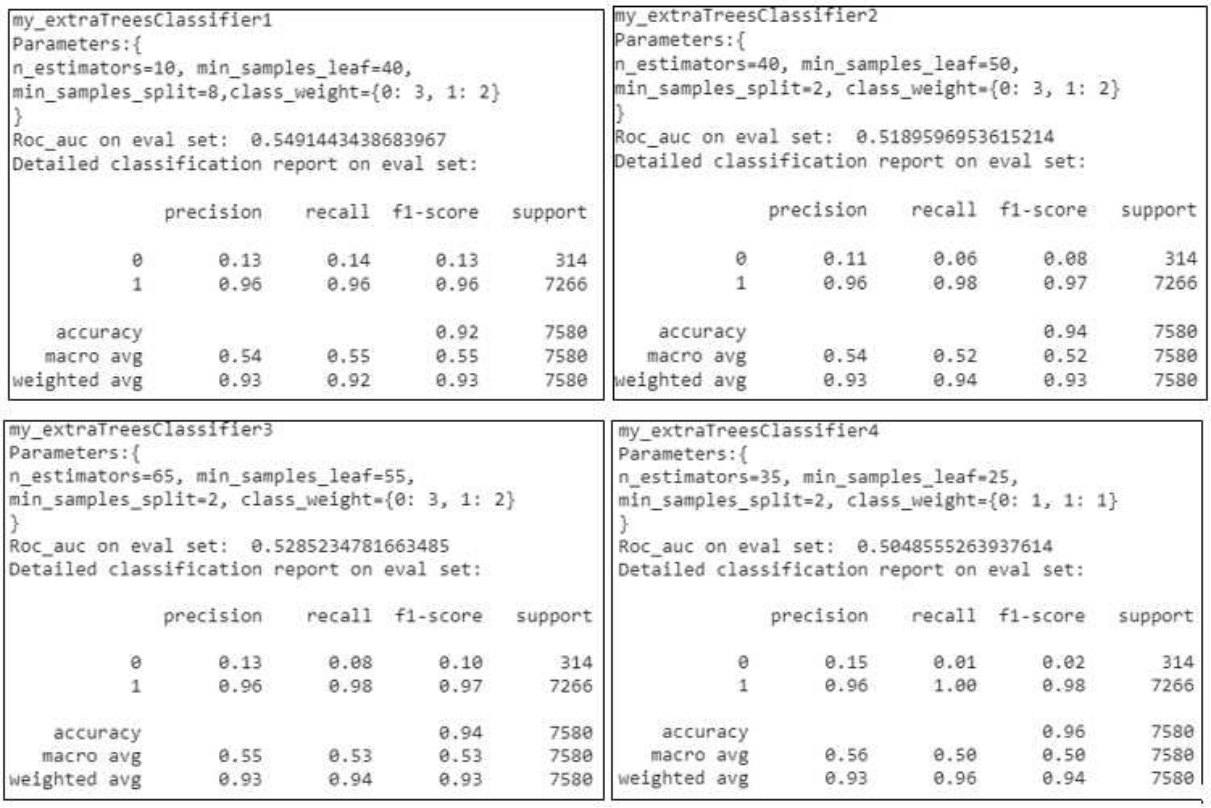
Proučavanjem skupa LCC-FASD uočeno je da su slike kojima je dio naziva *<model_uređaja>_<identifikator_osobe>_<scena>* jednak međusobno jako slične. Pretpostavka je da su uzorkovane iz istog videa s razmakom od 15-30 okvira. Testiranjem je zaključeno da korištenjem svih slika iz skupa za učenje i testnog skupa za učenje klasifikatora, klasifikator prenauči podatke i ima lošu sposobnost

generalizacije. Stoga su iz skupova za učenje i testiranje odabrane po dvije slike s jednakim spomenutim dijelom naziva, a ostale se nisu koristile. Klasifikatori učeni na ovako uzorkovanim podatcima imaju puno bolju sposobnost generalizacije nego klasifikatori učeni na svim podatcima iz skupova za učenje i test.

Slike skupa za evaluaciju nisu organizirane u poddirektorije, već se oznaka klase nalazi u nazivu slike. Nazivi slika imaju sljedeći format:
<oznaka_klase>_<redni_broj_slike>.png, kao na primjer *real_123.png* ili *spoof_0.png.*
Za evaluaciju naučenih modela korištene su sve slike iz skupa za evaluaciju.

Pretprocesiranje je obavljeno tako da je za svaku sliku izvršena pretvorba u YCrCb i CIE L*u*v prostor boja te se za svaku od komponenti ova dva prostora boja izradio histogram. Dobivenih šest histograma spojeno je u vektor značajki (engl. *feature vector*) $\mathrm{FV} = (\mathrm{Y}, \mathrm{Cr}, \mathrm{Cb}, \mathrm{L}, \mathrm{u}, \mathrm{v})$ veličine 1536 (šest normaliziranih histograma u rasponu od 0 do 255). Za svaku klasu podskupa kreirana je jedna datoteka koja sadrži numpy polje sa svim vektorima značajki slika koje pripadaju klasi unutar podskupa, kako je prikazano na desnoj strani Slika 4.14.
Za učenje modela formiran je skup podataka od vektora značajki slika iz originalnih podskupova za učenje i test uzorkovanih na temelju naziva kako je opisano iznad. Dobiveni skup je podijeljen u dio za učenje (70%) i dio za validaciju (30%), te su podatci nasumično izmiješani (engl. *shuffle*).

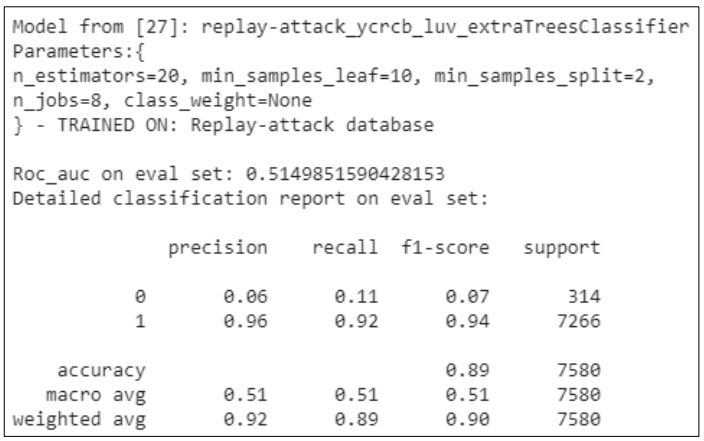
Na slici ispod prikazane su specifikacije i rezultati 4 klasifikatora s najboljim performansama na skupu za evaluaciju, nazvanih *my_extraTreeClassifier1, my_extraTreeClassifier2, my_extraTreeClassifier3 i my_extraTreeClassifier4.*
Na svakoj slici, ispod naziva navedeni su i parametri korišteni za izgradnju klasifikatora te rezultati na skupu za evaluaciju iz LCC-FASD skupa podataka izraženi mjerama preciznost (engl. *precision*), odziv (engl. *recall*), F1, specifičnost (engl. *specificity*) i površina ispod ROC krivulje (Roc_auc).



Slika 4.15: Rezultati evaluacije 4 različita ExtraTreesClassifier klasifikatora na podskupu za evaluaciju iz LCC-FASD

Usporedba s rezultatima iz literature

Naučeni model za detekciju prezentacijskih napada iz [27] preuzet je iz Github repozitorija jednog od autora rada [39]. Na slici ispod prikazane su specifikacije i rezultati tog klasifikatora na skupu za evaluaciju LCC-FASD.



Slika 4.16: Rezultati evaluacije modela iz [27] na podskupu za evaluaciju iz LCC-FASD

Iz rezultata na slikama Slika 4.15 i Slika 4.16 vidljivo je da sva 4 klasifikatora naučena u sklopu ovog rada imaju jednako dobre ili bolje rezultate od modela iz [27]. Međutim, budući da je taj klasifikator učen na skupu podataka Replay Attack, a ne na skupu podataka LCC-FASD na kojem su učeni klasifikatori *my_extraTreeClassifier 1-4*, obavljena je dodatna metoda evaluacije.

Prikupljen je mali skup podataka u svrhu evaluacije modela koji se sastoji od osobnih fotografija autorice ovog rada. Najprije je odabrano ukupno 26 različitih fotografija na kojima se nalazi po jedno lice (lice pripada jednom od ukupno 3 različita identiteta). Zatim je svaka od tih slika prikazana na ekranu mobilnog telefona te je web-kamerom laptopa snimljen ekran telefona. Svaka slika snimljena je 3 puta, na različitim udaljenostima od kamere laptopa. Na taj način prikupljen je skup lažnih slika koji se sastoji od ukupno 78 slika (26 x 3). Zatim je tom istom kamerom laptopa snimljeno dodatnih 15 slika stvarnog lica u blago različitih uvjetima osvjetljenja i one čine skup slika stvarnih lica. Skup za evaluaciju izgrađen je od vektora značajki prikupljenih slika (njih 93).

Klasifikatori *my_extraTreeClassifier 1-4*, kao i klasifikator iz [27] evaluirani su na opisanom skupu. Svaki klasifikator vraća vektor čiji su elementi vjerojatnost da je klasificirana slika stvarno lice i vjerojatnost da je klasificirana slika lažno lice. Napad je detektiran ako je izračunata vjerojatnost da je klasificirana slika lažno lice veća od nekog praga. Prilikom evaluacije, svaki klasifikator je evaluiran s obzirom na vrijednosti praga iz skupa: {0.60, 0.65, 0.70, 0.75, 0.80, 0.85}. Tablica prikazuje vrijednost praga za koju je svaki klasifikator imao najbolje rezultate pri evaluaciji kao i same rezultate evaluacije svih 5 klasifikatora.

	Prag	Roc_auc	TN (real_ok)	FP (real_nok)	TP (spoof_ok)	FN (spoof_nok)
my_extraTClassifier1	0.6	0.8352	11	4	72	6
my_extraTClassifier2	0.6	0.6250	7	8	73	5
my_extraTClassifier3	0.6	0.7159	7	8	73	5
my_extraTClassifier4	0.7	0.6420	7	8	62	16
extraTClassifier iz [27]	0.75	0.7848	13	2	54	24

Tablica 6: Evaluacija klasifikatora na privatnom skupu

4.4 Implemetirana metoda za raspoznavanje lica

Metoda za raspoznavanje lica implementirana u ovom radu inspirirana je znanstvenim člankom *ArcFace: Additive Angular Margin Loss for Deep Face Recognition* [20].

Naučena su 3 modela za verifikaciju lica s otvorenim skupom.
Implementacija modela temelji se na kodu iz službenog Github repozitorija navedenog u znanstvenom radu [20] i kodu iz Github repozitorija službenih reimplementacije projekta u Tensorflowu [37] i [41].

Arhitektura neuronske mreže sastoji s od bazne ResNet50 arhitekture na čiji je zadnji konvolucijski sloj dodana struktura slojeva BN-Dropout-FC-BN [20] za dobivanje ugrađenih vektora značajki (engl. *embeddings*). Korišten je ResNet50 model iz Keras biblioteke s prenaučenim ImageNet težinama da se smanji vrijeme učenja modela.

Modeli se razlikuju po slojevima gubitka. Prvi model (baseline) koristi softmax gubitak, drugi koristi A-Softmax gubitak, a treći ArcFace gubitak.
Svi modeli učeni su na MS1M-ArcFace skupu podataka, kako je predloženo u [20]. Skup sadrži 85 742 klase i ukupno 5 822 653 slike dimenzija (*112 x 112 x 3).*

Svi modeli učeni su 12 epoha. Korišteni optimizator je SGD s momentom 0.9.

Veličina mini-grupe (engl. *mini-batch size*) u implementaciji ovog projekta 256, za razliku od [20] gdje iznosi je 512. Broj iteracija u jednoj epohi iznosi 22744, i dobije se dijeljenjem ukupnog broja primjera u skupu za učenje s veličinom mini-grupe.

U bloku BN-Dropout-FC-BN dodanom na zadnji sloj ResNet50 mreže korištena je L2 regularizacija s faktorom smanjenja težina (engl. *weight decay*) $5 * 10^{-4}$.

Vrijednost parametra m za ArcFace gubitak iznosi 0.5, a za A-Softmax 1.35 [20].

Vrijednost stope učenja smanjuje se tijekom učenja kako je prikazano u tablici ispod.

Epoha	1-5	6-8	9-10	11-12
Vrijednost stope učenja	10^{-3}	$5*10^{-4}$	10^{-4}	$5*10^{-5}$

Tablica 7: Vrijednosti stope učenja modela za raspoznavanje lica

Naučeni modeli evaluirani su na varijanti LFW skupa podataka koja se sastoji od 6000 parova slika. Neki parovi se sastoje od dvije slike iste osobe, a neki od dvaju slika na kojima su lica različitih osoba. Za evaluaciju modela korištena je k-struka unakrsna provjera (engl. *k-folded cross validation*), gdje je k=10. Pseudokod algoritma implementiranog u ovom radu naveden je u nastavku.

```
ulazi:
        lfw # vektor dimenzija (12000, 3, 112, 112) - 12000 slika poredanih kao 6000 parova
        actual_issame # vektor dimenzija (6000,) koji na indeksu i sadrži 1 ako i-ti par slika u
                      # skupu lfw sadrži lica koja pripadaju istoj osobi, a 0 inače

embeddings = model (lfw) # izračunaj vektore značajki za lfw skup
                         # embeddings ima dimenzije (12000, 512)
embeddings1 = embeddings[0::2] # sadrži vektore prve slike svakog para; (6000,512)
embeddings2 = embeddings[1::2] # sadrži vektore druge slike svakog para; (6000,512)
dist = euclidean_distance (embeddings1, embeddings2)
nrof_pairs = 6000 # broj parova
indices = np.arange(nrof_pairs) # vrijednosti od 0 do nrof_pairs (0-6000)
thresholds = np.arange(0, 4, 0.01) # vrijednosti od 0 do 4 s korakom 0.01

# inicijalizacija generatora k preklopa
k_fold = sklearn.model_selection.KFold (n_splits=10 , shuffle=False)
# „učimo" klasifikator na k-1 preklopa i ispitujemo ga na k-tom preklopu
# to ponavljamo ukupno k-puta s pomicanjem ispitnog skupa
for fold_idx, (train_set, test_set) in enumerate(k_fold.split(indices)):
   # Nalaženje najboljeg praga za trenutni preklop
   acc_train = np.zeros((nrof_thresholds))
   for th_idx, threshold in enumerate(thresholds):
      acc_train[th_idx] = calc_accuracy(threshold, dist[train_set], actual_issame[train_set])
```

```
        best_th_index = np.argmax(acc_train)
    # zapamti najbolji prag za ovu iteraciju
    best_thresholds[fold_idx] = thresholds[best_th_index]
    # zapamti točnost na testnom skupu za ovu iteraciju
    accuracy[fold_idx] = calc_accuracy(thresholds[best_th_index], dist[test_set],
                                       actual_issame[test_set])

# vrati kao rezultat srednju vrijednost točnosti na testnim skupovima i
# srednju vrijednost najboljeg praga u svakoj iteraciji
return accuracy.mean(), best_thresholds.mean()
```

Rezultati evaluacije naučenih modela na skupu LFW prikazani su u tablici ispod.

Gubitak	Acc na LFW	Threshold
Softmax	85.12%	0.33
A-Softmax	93.15 %	0.12
ArcFace	98.37%	0.75

Usporedba sa službenim rezultatima (iz literature i službenih Github repozitorija)

U radu [20] na MS1M-ArcFace skupu podataka učen je model s baznom arhitekturom ResNet100. Model s baznom ResNet50 arhitekturom učen je na CASIA skupu podataka. Službeni git repozitorij [40] naveden u [20] u sekciji Model-Zoo[32] navodi točnost od 99.80 % modela s baznom arhitekturom ResNet50 učenog na MS1M-ArcFace skupu. Github repozitorij [37] navodi točnost od 99.35 % modela s baznom arhitekturom ResNet50 učenog na MS1M-ArcFace skupu.

Podatci o performansama modela s A-Softmax i softmax gubitkom i baznom arhitekturom ResNet50 nisu dostupni ni na jednom od spomenutih izvora.

[32] Adresa Model-Zoo [40]: https://github.com/deepinsight/insightface/wiki/Model-Zoo

Bez obzira na to, rezultati postignuti modelima implementiranim u sklopu ovog rada potvrđuju hipotezu rada [20], kao i njegovih prethodnika [19] i [21] da modeli s kutnom marginom (A-Softmax, ArcFace) imaju bolje performanse za problem raspoznavanja lica s otvorenim skupom od modela s marginom u euklidskom prostoru (softmax).
Nadalje, postignuti rezultati potvrđuju i tvrdnju da značajke naučene modelom sa softmax gubitkom nisu dovoljno diskriminativne za problem raspoznavanja lica s otvorenim skupom, iako su dovoljno odvojive za problem klasifikacije zatvorenog skupa.

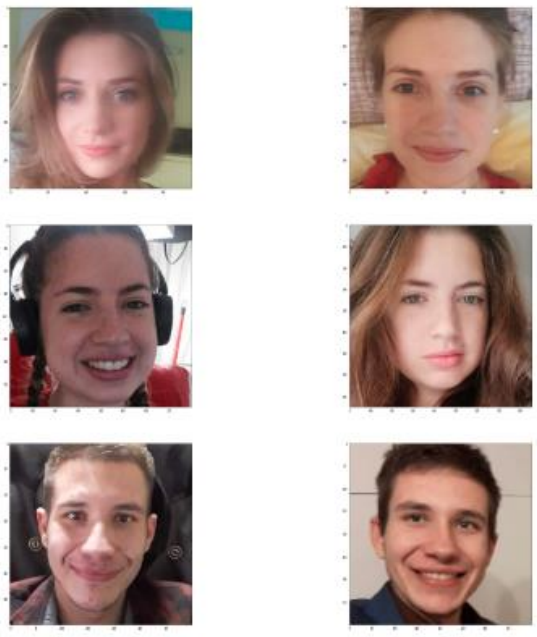
Za verifikaciju lica u izgrađenom sustavu za autentikaciju osoba analizom lica korišten je model s ArcFace gubitkom. Model ima točnost od 98.37% na skupu za evaluciju LFW. Na slikama ispod (Slika 4.17, Slika 4.18) prikazani su neki parovi lica skupa za evaluaciju na kojima je odabrani model griješio. Promatranjem tih parova vidljivo je da do pogreške najčešće dolazi u situacijama kada postoje velike varijacije u pozi i izrazima lica među slikama u paru, kao i u situacijama kada dodatni objekti poput naočala prekrivaju dio lica.

Slika 4.17: ResNet50 s ArcFace gubitkom - FN (eng. false negatives)

Slika 4.18: ResNet50 s ArcFace gubitkom - FP (eng. false positives)

Evaluacija na vlastitom skupu podataka

Prikupljen je mali skup podataka u svrhu dodatne evaluacije modela odabranog za korištenje u sklopu sustava za autentikaciju. Skup se sastoji od osobnih fotografija autorice ovog rada. Odabrano je ukupno 9 različitih slika na kojima se nalazi jedno lice (lice pripada jednom od ukupno 3 različita identiteta).



Slika 4.19: Primjeri slika vlastitog skupa za evaluaciju modela za raspoznavanje lica

Za svako lice izračunat je vektor značajki modelom s ArcFace gubitkom. Skup za evaluaciju izgrađen je od svih mogućih parova vektora značajki.

Evaluacija je obavljena k-strukom unakrsnom provjerom, gdje je k=10. Model je imao točnost od 100%, a vrijednost praga iznosila je 0.55.

Zatim je izgrađena baza vektora značajki lica autoriziranih korisnika koja će se koristiti u sklopu procesa autentikacije. Vektor koji se pohranjuje u bazu i koji će se kasnije koristiti za verifikaciju lica prilikom autentikacije, dobiven je kao srednja vrijednost vektora značajki lica koje pripadaju istom identitetu. Budući da sadrži samo tri vektora, baza je predstavljena npy datotekom koja sadrži vektore značajki lica pojedinog idetiteta. Pri verifikaciji lica u stvarnom vremenu, vektor značajki lica koje se verificira uspoređuje se sa svim vektorima iz baze.

4.5 Aplikacija za testiranje izgrađenog sustava za autentikaciju osoba analizom izgleda lica

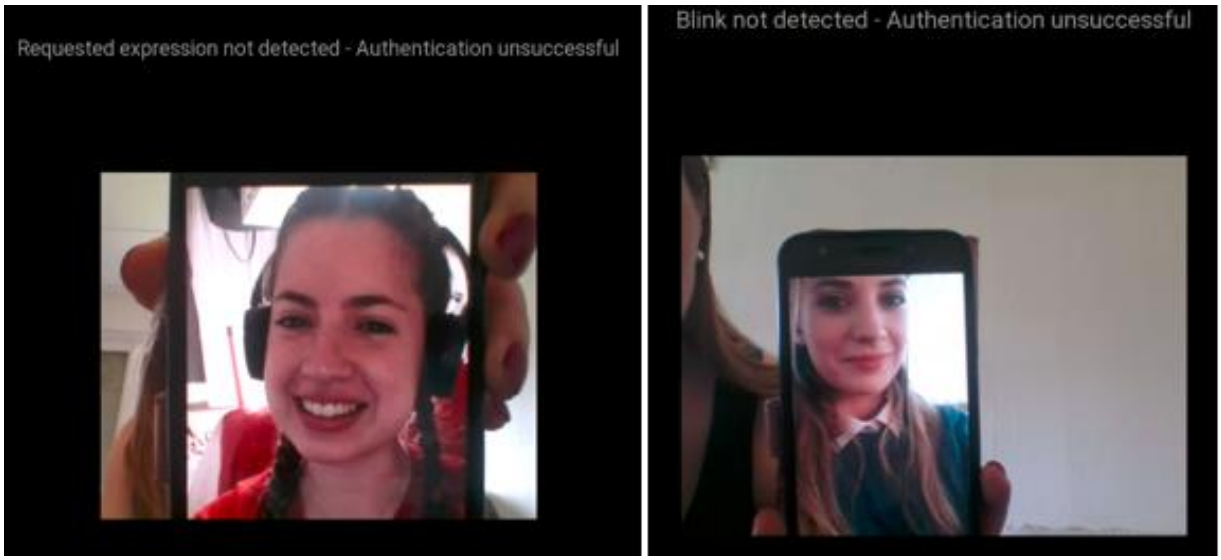
Za testiranje sustava izgrađena je desktop aplikacija za autentikaciju osobe u stvarnom vremenu. Tekst na korisničkom sučelju aplikacije pisan je na engleskom jeziku.

Nakon pokretanja procesa autentikacije upali se web-kamera te započinje proces detekcije prezentacijskih napada. Na ekranu se prikaže izazov (zahtjev) za korisnika i čeka se 2 sekunde da korisnik obavi traženi izazov.

Izazovi korišteni u aplikaciji su*:*

„Blink", „Smile", „Make surprised face", „Make angry face" i „Make sad face".

Ukoliko se unutar vremenskog intervala od dvije sekunde od trenutka prikaza zahtjeva ne detektira traženi odgovor od korisnika, sustav detektira napad, ispiše poruku o detekciji napada na ekranu aplikacije i prekida proces autentikacije. Ova metoda detekcije napada korisna je za detekcije 2D statičkih napada, kada se za napad koriste fotografije lica autorizirane osobe pri pokušaju napada na sustav.



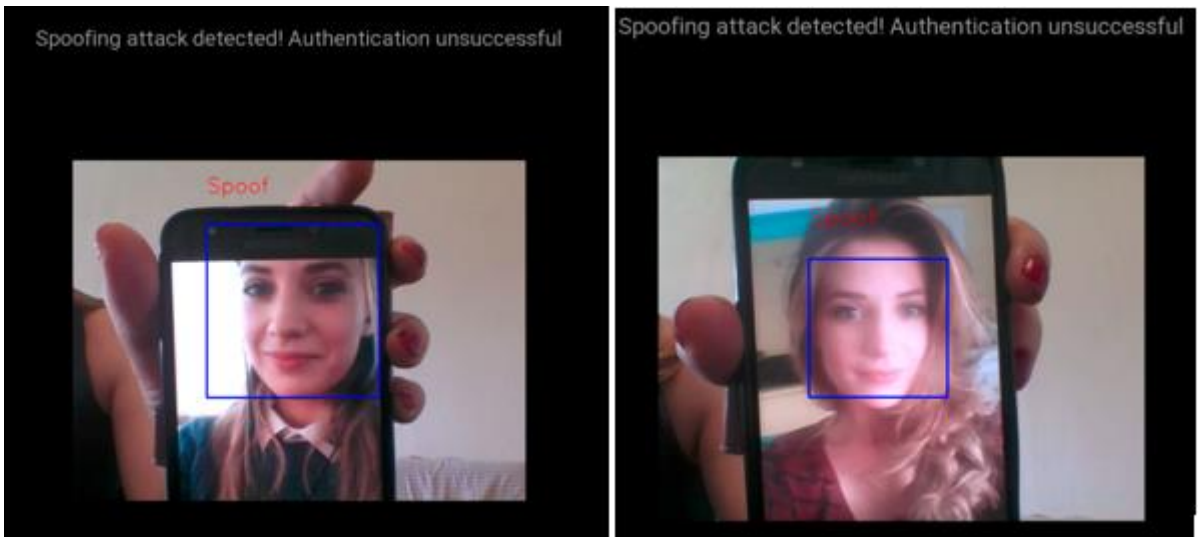
Slika 4.20: Primjer rada sustava za autentikaciju - uspješna detekcija napada - metoda izazov-odgovor

Budući da se izazov prikazan korisniku odabire nasumično, metoda uspješno detektira i neke dinamičke napade. Primjerice, ako se ova metoda detekcije napada želi

uspješno prevariti korištenjem videa koji sadrži lice autorizirane osobe, potrebno je nakon prikaza izazova na ekranu **unutar dvije sekunde** prikazati video sekvencu koja sadrži ispravan odgovor na traženi izazov.

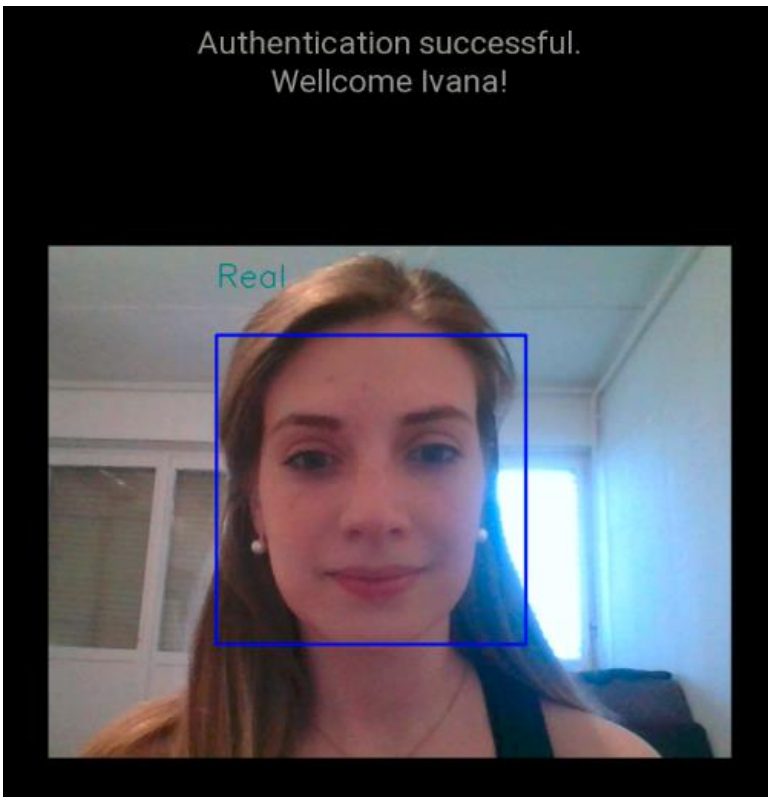
Za izvođenje takvog napada potrebno je poznavati sve izazove koje sustav može zatražiti i sukladno tome konstruirati ispravne video sekvence. Međutim, zbog vremenskog ograničenja na čekanje ispravnog odgovora od dvije sekunde, teško je izvesti napad na način da se konstruirani videi nalaze na uređaju (primjerice mobilnom telefonu) te se nakon prikaza izazova na ekranu aplikacije s telefona ručno pokrene video sekvenca sa ispravnim odgovorom. Za ovakve napade najčešće se koristi softver koji je sposoban detektirati traženi izazov iz slike ekrana aplikacije i na temelju detekcije automatski pokrenuti odgovarajuću video sekvencu.

Ukoliko se detektira ispravan odgovor na traženi izazov u zadanom vremenu, pokreće se druga faza detekcije dinamičkih 2D napada. Iz aktualnog okvira videa snimanog web-kamerom izreže se dio gdje je detektirano lice. Izrezani dio pretvara se u YCrCb i CIE L*u*v prostor boja te se za svaku od komponenti ova dva prostora boja računa histogram. Dobivenih šest histograma spaja se u vektor značajki koji se predaje *my_extraTreeClassifier1* klasifikatoru. Ukoliko se lice klasificira kao lažno, proces autentikacije se zaustavlja i ispisuje se poruka na ekranu aplikacije.



Slika 4.21: Primjer rada sustava za autentikaciju – uspješna detekcija napada - my_extraTreeClassifier1

Ukoliko se lice klasificira kao stvarno, izrezani dio slike koji sadrži lice predaje se modelu za raspoznavanje lica s ArcFace gubitkom koji računa vektor značajki za to lice. Zatim se taj vektor značajki uspoređuje sa svim vektorima značajki lica autoriziranih korisnika na način da se računa euklidska udaljenost između vektora. Ako je najmanja izračunata udaljenost manja od unaprijed definiranog praga (0.55), verifikacija lica je uspješno obavljena i na ekranu se ispisuje poruka dobrodošlice. Ako najmanja izračunata udaljenost nije manja od unaprijed definiranog praga, sustav zaključuje da osoba ispred kamere nije autorizirani korisnik te ispisuje prikladnu poruku na ekranu aplikacije.



Slika 4.22: Primjer rada sustava za autentikaciju – uspješna autentikacija nakon izazova „Smile“

Testiranje pomoću skupa videa

Za potrebe testiranja i ugađanja parametara sustava, snimljeno je 5 kratkih videa takvih da se na svakom izvodi po jedan od mogućih izazova (*„Blink”, „Smile”, „Make surprised face”, „Make angry face” i „Make sad face”)*.

Testiranje sustava na otpornost na dinamičke 2D prezentacijske napade izvršeno je prikazivanjem snimljenih videa na ekranu mobilnog uređaja. Nakon zadavanja izazova na ekranu aplikacije, odabran je video s očekivani odgovorom i ekran mobitela na kojem se video prikazuje okrenut prema web-kameri laptopa na kojem je pokrenuta aplikacija za autentikaciju.

Inicijalno vremenski interval tokom kojeg sustav čeka na odgovor na izazov bio je postavljen na 4 sekunde. Testiranjem je utvrđeno da je to dovoljno vremena da se uspješno izvede napad. Vremenski interval je zatim smanjen na 2 sekunde, što se pokazalo efikasnim u sprječavanju opisanog dinamičkog napada, ali dovoljno vremena da legitimni korisnik izvršni traženi izazov.

Vrste napada koje su sposobne generirati i prikazati kameri očekivani odgovor na traženi izazov najčešće se temelje na softveru za lokalizaciju i obradu teksta koji je sposoban u stvarnom vremenu snimiti ekran aplikacije na kojem je prikazan izazov, raspoznati tekst izazova i na temelju toga generirati video sekvencu ili sliku s licem autorizirane osobe koja obavlja traženi izazov. Otpornost sustava na ovakve vrste dinamičkih napada nije testirana, ali bi takvi napadi uspješno zaobišli vremensko ograničenje od 2 sekunde te bi uspješnost detekcije napada ovisila o klasifikatoru *my_extraTreeClassifier1.*

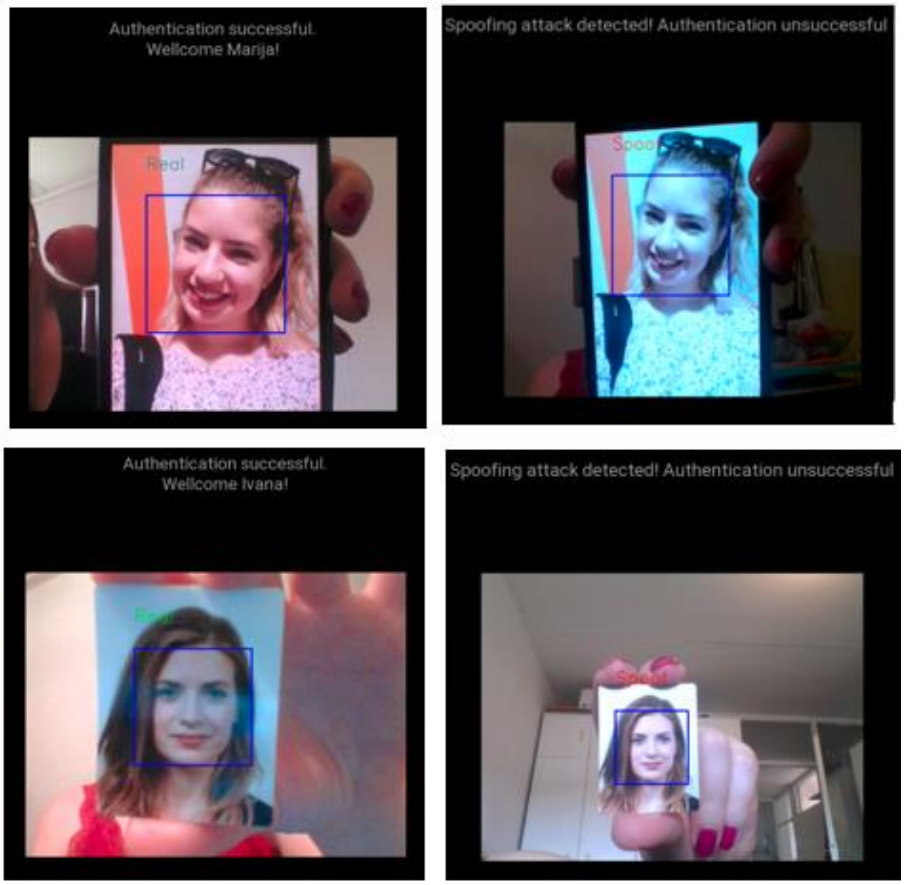
Testiranje pomoću skupa slika

Prikupljen je mali skup podataka u svrhu testiranja izgrađenog sustava za autentikaciju koji se sastoji od 25 osobnih fotografija autorice ovog rada.

Testiranje sustava na otpornost na statičke 2D prezentacijske napade izvršeno je prikazivanjem fotografija iz tog skupa za testiranje web-kameri laptopa na kojem je pokrenuta aplikacija za autentikaciju. Slika 4.20 i Slika 4.21 prikazuju uspješne detekcije izvedenih statičkih 2D napada. **Detektor izraza lica i detektor treptaja uspješno detektiraju sve napade u kojima se ne detektira očekivani odgovor na traženi izazov u vremenskom intervalu od dvije sekunde.**

U situacijama kada je kameri prikazano lažno lice koje zadovoljava traženi izazov (primjerice, ako je izazov glasio „Smile“ i kameri se prikaže fotografija nasmiješene osobe), pokreće se detekcija napada pomoću klasifikatora *my_extraTreeClassifier1*.

Tijekom testiranja primjećeno je da je klasifikator osjetljiv na promjene u okolnom osvjetljenju, kao što je prikazano na Slika 4.23.



Slika 4.23: my_extraTreeClassifier1 - pogreška klasifikacije

U procesu testiranja pomoću skupa slika primjećeno je da su pogreške *my_extraTreeClassifier1* klasifikatora primarni razlog koji dovodi do autentikacije neautoriziranih korisnika (Slika 4.23, lijevo). Zbog tog se korak detekcije dinamičkih 2D napada smatra kritičnom točku sustava. Da bi se smanjila vjerojatnost pogreške klasifikatora, za konačni rezultat klasifikacije uzima se prosjek rezultata na 10 uzastopnih okvira videa.

Model za raspoznavanje lica uspješno je obavio verifikaciju lica svaki put. Osjetljivost modela na varijacije u pozi i izrazima lica, koja je primjećena pri evaluaciji na LFW skupu podataka, nije došla do izražaja tokom procesa verifikacije lica u sklopu aplikacije.

Zaključak

U sklopu ovog rada proučene su arhitekture konvolucijskih mreža i metode za sprječavanje prezentacijskih napada. Izgrađen je funkcionalan sustav za autentikaciju lica koji se sastoji od komponenti za detekciju lica, detekciju treptaja, raspoznavanje ekspresija lica, klasifikatora *ExtraTreesClassifier* te raspoznavanje lica na otvorenom skupu.
Prikazani su i ocijenjeni rezultati svih implementiranih modela te je provedena detaljna usporedba s rezultatima iz literature.
Sve komponente povezane su u sustav za autentikaciju te je, zajedno s korisničkim sučeljem implementiranim s pomoću Kivy biblioteke, predstavljena aplikacija za autentikaciju osoba u stvarnom vremenu.

Metodom izazov-odgovor sustav uspješno detektira statičke 2D prezentacijske napade, a vremensko ograničenje čekanja na odgovor korisnika znatno otežava i izvođenje dinamičkih 2D napada.

Model za raspoznavanje lica uspješno je naučen na skupu MS1M-ArcFace i evaluiran na skup LFW, kao i na privatnom skupu. U sklopu sustava za autentikaciju uspješno obavlja verifikaciju autoriziranih korisnika, kao i detekciju neautoriziranih lica.
Klasifikator *ExtraTreesClassifier* predstavlja dodatnu zaštitu protiv statičkih i dinamičkih napada. Međutim, zbog osjetljivosti na promjene pozadinskog osvjetljenja predstavlja kritičnu točku sustava i uvodi stupanj nesigurnosti u rezultat cjelokupne klasifikacije. Implementacijom neke druge metode za detekciju dinamičkih napada može se poboljšati sigurnosna razina sustava predstavljenog u ovom radu. Također, za primjenu izrađene aplikacije na situacije kada je broj autoriziranih korisnika jako velik, bilo bi potrebno izmijeniti implementaciju baze vektora značajki autoriziranih korisnika i paralelizirati proces usporedbe ispitnog vektora značajki lica s onima u bazi. Još jedna dobra metoda bila bi naučiti klasifikator, primjerice SVM, da klasificira vektore značajki lica u klase koje odgovaraju identitetu osobe. Na taj bi se način izbjegla potreba za uspoređivanjem ispitnog vektora sa svim vektorima u bazi.

Literatura

[1] Jain A., Bolle R., Pankanti S. *Biometrics: Introduction to Biometrics*. Boston, MA: Springer, 1996.

[2] Weng J.J., Swets D.L. *Biometrics : Face Recognition*. Boston, MA: Springer, 1996.

[3] D. Timoshenko, K. Simonchik, V. Shutov, P. Zhelezneva and V. Grishkin. *Large Crowdcollected Facial Anti-Spoofing Dataset*. Computer Science and Information Technologies (CSIT), Yerevan, Armenia (201, str. 123-126.

[4] Simonyan K., Zisserman A. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* arXiv preprint arXiv:1409.1556, 2014.

[5] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. *Going Deeper with Convolutions.* arXiv preprint arXiv: 1409.4842, 2014.

[6] He K., Zhang X., Ren S., Sun J. *Deep Residual Learning for Image Recognition.* arXiv preprint arXiv:1512.03385, 2015.

[7] Chollet F. *Xception: Deep Learning with Depthwise Separable Convolutions.* arXiv preprint arXiv:1610.02357, 2016.

[8] Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* arXiv preprint arXiv:1704.04861, 2017.

[9] Adit Deshpande. The 9 *Deep Learning Papers You Need To Know About (Understanding CNNs Part 3),* GitHub (24.8.2016). Poveznica:

*https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.*html; pristupljeno 7. travnja 2020.

[10] Josef Burger. *A basic introduction to neural networks,* University Of Wisconsin–Madison. Poveznica: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html; pristupljeno 5.travnja 2020.

[11] Josip Žalac. *Primjena duboke neuronske mreže u raspoznavanju objekata*. Diplomski rad. Sveučilište Josipa Jurja Strossmayera u Osijeku Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, 2019.

[12] Hornik K., Stinchcombe M., White H. *Multilayer feedforward networks are universal approximators*. Neural networks str. 359 - 336. 1989.

[13] LeCun Y., Bottou L., Bengio Y., Haffner P. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[14] LeCun Y., Bengio Y. *The handbook of brain theory and neural networks : Convolutional networks for images, speech, and time series*, 1.izdanje. *Cambridge, Massachusetts* : MIT Press 1995

[15] Damjan Miko. *Duboki konvolucijski modeli za lokalizaciju objekata*. Diplomski rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2018.

[16] Vedran Vukotić. *Raspoznavanje objekata dubokim neuronskim mrežama*. Diplomski rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2014.

[17] *2D Average pooling*. Poveznica: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-average-pooling; pristupljeno 3.svibnja 2020.

[18] Liu W., Jiao J., Mo Y., Jiao J., Deng Z. *MaaFace: Multiplicative and Additive Angular Margin Loss for Deep Face Recognition.* Proceedings of 10th International Conference, ICIG 2019, Beijing, Kina, (2019), str. 642-653.

[19] Liu W., Wen Y., Yu Z., Li M., Raj B., Song L. *SphereFace: Deep Hypersphere Embedding for Face Recognition.* arXiv preprint arXiv:1704.08063, 2017.

[20] Deng J., Guo J., Xue N., Zafeiriou S. *ArcFace: Additive Angular Margin Loss for Deep Face Recognition*. arXiv preprint arXiv:1801.07698, 2018.

[21] Wang H., Wang Y., Zhou Z., Ji X., Gong D., Zhou J., Li Z., Liu W. *CosFace: Large Margin Cosine Loss for Deep Face Recognition.* arXiv preprint arXiv:1801.09414, 2018.

[22] Lin M., Chen Q., Yan S. *Network In Network.* arXiv preprint arXiv:1312.4400, 2013.

[23] *ArcFace: Additive Angular Margin Loss for Deep Face Recognition.* Poveznica: https://www.groundai.com/project/arcface-additive-angular-margin-loss-for-deep-face-recognition/1, pristupljeno 13.svibnja 2020.

[24] Tang D., Zhou Z., Zhang Y., Zhang K. *Face Flashing: a Secure Liveness Detection Protocol based on Light Reflections*. arXiv preprint arXiv:1801.01949, 2018.

[25] Wagner M., Chetty G. *Encyclopedia of Biometrics: Anti-spoofing, Face*. Boston, MA: Springer, 2015.

[26] Serhii Maksymenko. *Anti-Spoofing Techniques For Face Recognition Solutions,* TowardsDataScience, (6.8.2019). Poveznica: https://towardsdatascience.com/anti-spoofing-techniques-for-face-recognition-solutions-4257c5b1dfc9; pristupljeno 17.3.2020.

[27] Costa V., Sousa A., Reis A. *Image-Based Object Spoofing Detection.* Proceedings of 19th International Workshop, IWCIA 2018, Porto, Portugal, (2018), str. 189-201.

[28] Anjos A., Marcel S*. Counter-Measures to Photo Attacks in Face Recognition: a public database and a baseline.* Proceedings of International Joint Conference on Biometrics 2011, Washington, DC (2011), str. 1-7.

[29] Chingovska I., Anjos A., Marcel S. *On the Effectiveness of Local Binary Patterns in Face Anti-spoofing.* Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG), Darmstadt (2012), str. 1-7.

[30] Čehovin L., Mandeljc R. *Real-Time Eye Blink Detection using Facial Landmarks*. Proceedings of the 21st Computer Vision Winter Workshop, Rimske toplice, Slovenija (2016).

[31] Arriaga O., Valdenegro-Toro M., *Plöger P. Real-time Convolutional Neural Networks for Emotion and Gender Classification*. arXiv preprint arXiv:1710.07557, 2017.

[32] Guo Y., Zhang L., Hu Y., He X., Gao J*. MS-Celeb-1M: {A} Dataset and Benchmark for Large-Scale Face Recognition.* arXiv preprint arXiv: 1607.08221 2016.

[33] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments.* Technical report, 2007.

[34] Goodfellow I.J., Erhan D., Carrier P.L., Courville A., Mirza M., Hamner B., Cukierski W., Tang Y., Thaler D., Lee D.H., Zhou Y., Ramaiah C., Feng F., Li R., Wang X., Athanasakis D., Shawe-Taylor J., Milakov M., Park J., Ionescu R., Popescu M., Grozea C., Bergstra J., Xie J., Romaszko L., Xu B., Chuang Z., Bengio Y. *Challenges in Representation Learning: A report on three machine learning contests.* arXiv preprint arXiv: 1307.0414. 2013.

[35] Edmunds, Taiamiti & Caplier, Alice. (2018). *Face spoofing detection based on colour distortions.* IET Biometrics. 7,1 (2018), str. 27-38.

[36] Rosebrock A. *Eye blink detection with OpenCV, Python, and dlib*, PyImageSearch (24.4.2017). Poveznica: https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/, pristupljeno: 17.5.2020.

[37] Kuan-Yu Huang. peteryuX. *peteryuX/arcface-tf2*. 2019. Poveznica: https://github.com/peteryuX/arcface-tf2

[38] Octavio Arriaga. oarriaga. *Oarriaga/face_classification*. Poveznica: https://github.com/oarriaga/face_classification

[39] Valter Costa. ee09115. *ee09115/spoofing_detection.* Poveznica: https://github.com/ee09115/spoofing_detection

[40] DeepInsight. deepinsight. *deepinsight/insightface*. Poveznica: https://github.com/deepinsight/insightface/tree/master/src

[41] luckycallor. *luckycallor/InsightFace-tensorflow*. Poveznica: https://github.com/luckycallor/InsightFace-tensorflow

Sažetak

U sklopu ovog rada proučene su i opisane arhitekture konvolucijskih neuronskih mreža i funkcije gubitka za problem raspoznavanje lica i prepoznavanja izraza lica. Izgrađen je sustav za autentikaciju lica koji se sastoji od komponenti za detekciju lica, detekciju treptaja, modela za raspoznavanje ekspresija lica učenog na FER2013 skupu podataka te evaluiranom na CK+48 skupu podataka, klasifikatora *ExtraTreesClassifier* učenog na LCC-FASD skupu podataka i modela za raspoznavanje lica na otvorenom skupu učenog na MS1M-ArcFace skupu podataka i evaluiranom na LFW skupu podataka.

Prikazani su i ocijenjeni rezultati svih implementiranih modela, prikazani pogrešno klasificirani primjeri te je provedena detaljna usporedba s rezultatima iz literature.

Sve komponente povezane su u sustav za autentikaciju te je, zajedno s korisničkim sučeljem implementiranim pomoću Kivy biblioteke, predstavljena funkcionalna aplikacija za autentikaciju osoba u stvarnom vremenu.

Ključne riječi: neuronske mreže, duboke neuronske mreže, konvolucijske neuronske mreže, raspoznavanje lica, računalni vid, duboko učenje, ResNet, ResNet50, Xception, Mini_Xception, ExtraTreesClassifier, ArcFace, SphereFace, MSM1-Arcface, LFW, FER2013, CK+48, LCC-FASD, raspoznavanje izraza lica, detekcija treptanja, prezentacijski napadi.

Summary

In this work, the specific architectures and loss functions for deep neural networks are shown with an emphasis on convolutional neural networks for face recognition and facial expression recognition. A system for face authentication has been implemented, containing components for face detection, blink detection, model for face expression detection trained on FER2013 dataset and evaluated on CK+48 dataset, *ExtraTreesClassifier* trained on LCC-FASD dataset and model for face recognition trained on MS1M-ArcFace dataset and evaluated on LFW dataset.

Furthermore, numerical results of all implemented models are expressed, their performance is evaluated and compared with the results from the literature and misclassified examples are briefly discussed.
All mentioned components are connected together to form a functional real-time face authentication system with user friendly interface implemented using Kivy framework.

Keywords: neural networks, deep neural networks, convolutional neural networks, face recognition, computer vision, deep learning, ResNet, ResNet50, Xception, Mini_Xception, ExtraTreesClassifier, ArcFace, SphereFace, MSM1-Arcface, LFW, FER2013, CK+48, LCC-FASD, facial expression recognition, blink detection, presentation attacks.